UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

APLICACIÓN DE LA METODOLOGÍA POWERDIGIT AL DISEÑO DEL CONTROL DIGITAL DE UN CONVERTIDOR DE POTENCIA

PABLO IGLESIAS GIL

TUTORAS

CLARA MARINA SANZ GARCÍA

MARTA PORTELA GARCÍA

Leganés, Septiembre 2015

A mis tutoras, Marina y Marta, por darme esta oportunidad y por el apoyo que he recibido en todo momento.

A mi padre, mi héroe, tu ausencia me ha obligado a ser más fuerte; tu recuerdo alimenta mi esperanza. Tus enseñanzas, son mi tesoro.

A Ali, mi compañera de viaje, mi mayor orgullo, por todo tu amor y confianza. Sin ti nada de esto hubiera sido posible.

Gracias.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

ADC.	ANALOG TO DIGITAL CONVERTER.
CC.	CORRIENTE CONTINUA.
DAC.	DIGITAL TO ANALOG CONVERTER.
DCM.	DIGITAL CLOCK MANAGER.
DPWM.	DIGITAL PULSE-WIDTH MODULATION.
DSP.	DIGITAL SIGNAL PROCESSOR.
FPGA.	FIELD PROGRAMMABLE GATE ARRAY.
HDL.	HARDWARE DESCRIPTION LANGUAGE.
IP.	INTELLECTUAL PROPERTY.
MF.	MARGEN DE FASE.
MOSFET.	METAL-OXIDE-SEMICONDUCTOR FIELD-EFFECT TRANSISTOR.
PCB.	PRINTED CIRCUIT BOARD.
PWM.	PULSE-WIDTH MODULATION.
RTL.	REGISTER TRANSFER LANGUAGE.
SPI.	SERIAL PERIPHERAL INTERFACE.
TCL.	TOOL COMMAND LANGUAGE.
VHDL.	VHSIC-HDL
VHSIC.	VERY HIGH SPEED INTEGRATED CIRCUIT.

1. INTRODUCCIÓN

Los convertidores de potencia utilizan dispositivos para acondicionar la salida, en tensión o corriente, con el objetivo de cumplir las especificaciones del sistema. Estos dispositivos requieren señales de control que permitan asegurar la estabilidad del convertidor ante cualquier tipo de perturbación. Para realizar estas tareas de control es necesario implementar un sistema realimentado que cuente con al menos un regulador. En ocasiones, será necesario incorporar un modulador de señal e incluso, varios reguladores.

En el caso de los convertidores CC-CC, hasta hace pocos años, la mayoría de los reguladores se diseñaban con componentes analógicos. Sin embargo, para gran parte de las aplicaciones en las que se emplean convertidores existen numerosas ventajas de implementar reguladores digitales frente a los analógicos convencionales [1]. Algunas de las cuales son:

- Es posible implementar algoritmos de control más complejos siendo capaces de mejorar las características estáticas y dinámicas del sistema [2].

- El sistema tendrá una mayor flexibilidad ya que los dispositivos digitales tales como FPGAs o microprocesadores son reprogramables, siendo los parámetros de regulación fácilmente modificables [3].

- Los tiempos de diseño se reducen debido a la simplicidad de realizar cambios sobre un prototipo [4].

- Mejora de la fiabilidad del sistema debido a la reducción de componentes discretos como resistencias y condensadores, lo que permite disminuir la complejidad del control ante necesidad de sistemas complejos [3] [5].

- Evitamos problemas de funcionamiento debidos al envejecimiento de los componentes, muy acusados en condensadores, y además hacemos a los sistemas más independientes de la temperatura [2].

- Mayor inmunidad frente al ruido [2].

- Fácil integración a sistemas digitales más complejos [3] [4].

A pesar de todas las ventajas citadas también existen algunos inconvenientes derivados de este tipo de implementación:

- Se requiere añadir un conversor analógico digital como nuevo componente del sistema [5].

- Existe una limitación en la resolución de medida puesto que los datos se representan utilizando un número determinado de bits [5].

- El dispositivo digital empleado limita la resolución de la señal de control, truncando así el valor final [6].

- Los sistemas digitales introducen retardos respecto a la implementación analógica que deberán ser tenidos en cuenta por el diseñador. Los retrasos más importantes son fundamentalmente el introducido por el ADC al realizar la conversión digital y el regulador durante el tiempo de cálculo de la señal de control [6].

Finalmente, el diagrama de bloques típico de un sistema de control digital aplicado a convertidores de potencia se muestra en la figura 1:

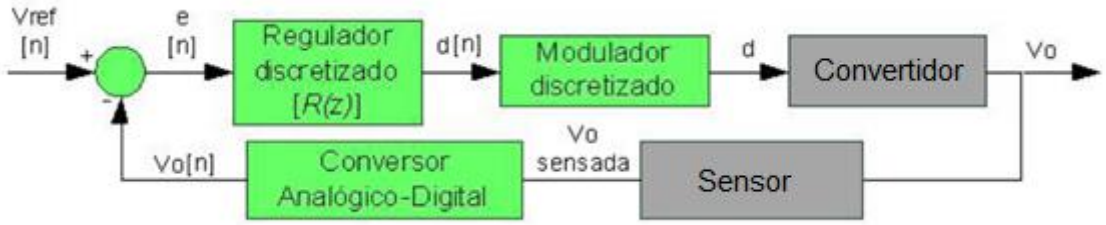


FIGURA 1. DIAGRAMA DE BLOQUES DE UN SISTEMA REALIMENTADO DIGITAL

A diferencia de los lazos de control analógicos, este sistema requiere de un conversor analógico-digital (ADC) para poder introducir el dato medido en el dispositivo digital. El resto de bloques serán comunes tanto para diseños de control analógicos como para diseños digitales. Otro aspecto a tener en cuenta es que no es necesario incorporar un conversor digital-analógico (DAC) al sistema digital puesto que para el caso de los convertidores, la señal de control que dispara los dispositivos de la etapa de potencia es directamente la señal obtenida por el modulador.

Ya que en estos sistemas se involucran diseños digitales y de convertidores de potencia, el presente trabajo ha sido desarrollado en el Departamento de Tecnología Electrónica con la colaboración de los grupos de investigación de Sistemas Electrónicos de Potencia y de Diseño Microelectrónico y Aplicaciones.

1.1. Motivación

Cuando el diseñador decide implementar un control digital para este tipo de sistemas lo realizará mediante algún lenguaje de descripción hardware, como VHDL o Verilog. Una vez se haya finalizado el diseño se podrán realizar pruebas de los bloques, pero al ser todas ellas en lazo abierto no nos aseguran que el diseño sea adecuado. Por ello, para validar el sistema en lazo cerrado podemos recurrir a co-simulación, si utilizamos dispositivos reprogramables como FPGAs, mediante el cual nos es posible simular la parte analógica y la digital de forma conjunta empleando las herramientas de PSIM y ModelSim.

Si el resultado obtenido tras la co-simulación no es el esperado, la depuración del sistema es complicada. No se dispone de un diseño de referencia con el que comparar los resultados obtenidos, dificultando la tarea de acotar los posibles fallos; bien sea por una inadecuada elección del número de bits en la representación de datos, lo que provocaría imprecisión, o un fallo a la

hora de diseñar el código. En el caso de otro tipo de plataformas digitales como los microprocesadores o los DSPs no es posible co-simular, por lo que contar con un diseño de referencia que reproduzca fielmente la implementación digital se antoja necesario.

La metodología PowerDigit [1] pretende solucionar estos problemas derivados de la implementación digital para reducir el tiempo de diseño y validación del sistema, y mejorar, en definitiva, las herramientas de las que dispondrá el diseñador. Para ello, la metodología propone un sistema funcional de referencia a partir del cual se diseñará el hardware, y que permite disponer de un modelo comportamental con el que contrastar los resultados de co-simulación. Además, proporciona una serie de procedimientos y normas a cumplir cuando se definen los parámetros del lazo y que permiten garantizar el correcto diseño del sistema.

Sin embargo, el estado actual de la metodología PowerDigit plantea una serie de limitaciones:

- En primer lugar, el sistema en bloques funcionales sólo dispone de un modelo de ADC convencional, siendo imposible emplear otro tipo de ADC sin tener consecuencias directas en la implementación hardware.
- El diseño del lazo sólo puede llevarse a cabo en la herramienta de simulación PSIM, lo que supone una barrera para diseñadores que emplean otro tipo de herramientas.
- Por otro lado, el diseño no cuenta con un generador automático de código a partir de plantillas predefinidas, lo que obliga al usuario a introducir modificaciones directamente en el código VHDL.
- Por último, no se han realizado validaciones experimentales de diseños que aplican esta metodología. Lo cual implica que aún no se ha cerrado la fase de diseño completamente.

El presente trabajo surge para abordar todas estas limitaciones con el fin de mejorar la metodología PowerDigit y extenderla a otro tipo de modelos y otro tipo de herramientas. Además de cerrar el ciclo de diseño realizando pruebas experimentales.

Por último, se pretende dar una visión práctica de aplicación de la metodología que permita al lector realizar sus propios diseños sin necesidad de conocimientos profundos en sistemas digitales.

1.2. Objetivos

Como ya se ha comentado en el apartado 1.1. El objetivo principal de este proyecto es extender y aplicar la metodología PowerDigit en el diseño de un lazo de control digital para un convertidor CC-CC y realizar las validaciones necesarias tanto en co-simulación como experimentalmente del sistema completo. Con el fin de eliminar las limitaciones citadas anteriormente se han propuesto los siguientes objetivos para el presente trabajo:

- Diseño de los bloques funcionales empleando MATLAB.
- Desarrollo de nuevos modelos de ADC para bloques funcionales.
- Configuración automática del código VHDL a través de plantillas.

- Modificación de la descripción del circuito digital VHDL existente en [1] para incluir las mejoras desarrolladas en este trabajo y generalizarlo.

- Validación experimental del sistema completo.

Para llevar a cabo estos objetivos se realizarán una serie de tareas cuya descripción detallada puede verse en el capítulo 7 del presente trabajo.

1.3. Herramientas empleadas

Durante el desarrollo del trabajo se han empleado diversas herramientas de simulación y diseño, las cuáles se citan a continuación:

- **PSIM**. Es una herramienta de simulación de circuitos eléctricos y electrónicos de interfaz similar a otros programas más populares como Pspice y optimizado para sistemas de potencia. Además cuenta con una amplia variedad de herramientas para el diseño de sistemas de control, así como un módulo de co-simulación de circuitos de señal mixta llamado ModCoupler.

- **SmartCtrl**. Esta herramienta permite diseñar lazos de control analógicos de manera rápida y sencilla. Pudiendo obtener gráficamente información estática y dinámica del sistema. Además permite diseñar lazos digitales simples, sin tener en cuenta ninguna de las restricciones que este tipo de sistemas imponen.

- **ModelSim**. Es una herramienta de simulación de sistemas digitales que cuenta con la posibilidad de admitir lenguajes tanto de hardware como de script. Para el presente trabajo servirá como interfaz para visualizar las señales intermedias implementadas y poder depurar el sistema en caso de fallo.

- **Xilinx ISE Design Suite**. Herramienta de diseño, simulación y síntesis de sistemas digitales. Permite diseñar sistemas hardware así como realizar los test necesarios y posteriormente la síntesis y Layout del código diseñado para implementarlo en la FPGA. Por último también servirá de interfaz entre la FPGA y el PC para grabar el código.

- **MATLAB/Simulink**. Esta herramienta de cálculo además permite simular toda clase de sistemas (mecánicos, eléctricos, etc.), se empleará para comparar los diseños de PSIM y de la propia herramienta Simulink, además se utiliza la toolbox de *SimPowerSystem* para representar la planta de nuestro sistema.

1.4. Estructura del documento

La estructura que seguirá el presente trabajo se detalla a continuación:

En primer lugar el capítulo 1 contiene una introducción en la cual se desarrollan las ideas más importantes de diseñar sistemas de control digital, mencionando la problemática existente;

además se expone la motivación para llevar a cabo este trabajo así como los objetivos que se pretenden lograr.

El capítulo 2 presenta una breve introducción del trabajo previo realizado donde se hará especial énfasis en todos aquellos puntos que servirán de partida para desarrollar este trabajo así como los puntos que serán necesarios para aplicar la metodología PowerDigit con éxito.

En el capítulo 3 se propone la extensión del estado actual incorporando otros modelos de ADC, empleando otras herramientas de simulación más extendidas para realizar el diseño del lazo en bloques funcionales, incluso la posibilidad de generar código automáticamente a través de estas herramientas. Otro punto de este capítulo tratará de explicar las modificaciones llevadas a cabo en la descripción del circuito digital.

El capítulo 4 presenta las características de los distintos elementos utilizados en el trabajo, como son el convertidor CC-CC y la placa FPGA. Se enumerarán las especificaciones que debe cumplir el sistema y por último se realizarán paso a paso todos los cálculos necesarios para este diseño del lazo según la metodología PowerDigit.

El capítulo 5 presenta todos los resultados obtenidos en el presente trabajo. Se divide en dos partes: En primer lugar se realiza una comparativa en profundidad entre los diseños en bloques funcionales de PSIM/MATLAB y el sistema de co-simulación, y por otro lado se realizan las pruebas experimentales relativas al sistema físico.

En el capítulo 6 se exponen las conclusiones del desarrollo de este trabajo así como las líneas de trabajo futuras a desarrollar.

En el capítulo 7 se detalla el presupuesto relativo a la consecución del proyecto así como un diagrama de planificación de las tareas llevadas a cabo en el mismo. Por último, existe un apartado de anexos en el que se desarrolla un breve manual de la herramienta de co-simulación, con el objetivo de paliar la escasa información acerca de su uso.

2. TRABAJO PREVIO

Como ya se comentó en el capítulo anterior, existen algunas consideraciones importantes a tener en cuenta a la hora de diseñar un lazo de control digital. Además, aparecen problemas en fases posteriores del diseño cuando se intenta depurar el sistema, puesto que no se dispone de una referencia para contrastar los resultados y detectar, e incluso prever, posibles fallos.

2.1. Consideraciones al diseñar el lazo de control digital

La metodología PowerDigit propone modificaciones y aportaciones al estado de la técnica actual teniendo en cuenta aspectos de la implementación digital tales como el número de bits de representación, los retrasos introducidos por los componentes digitales y las consideraciones acerca del redondeo en las operaciones. Las fases que incorpora esta metodología se resumen en los siguientes puntos [1]:

1. Elección del número de bits del DPWM y del ADC.
2. Cálculo de la ganancia del lazo.
3. Cálculo de los retrasos debidos a la implementación digital.
4. Obtención de los coeficientes del regulador.
5. Escalado de los coeficientes.
6. Diseño en bloques funcionales teniendo en cuenta la implementación hardware.
7. Validación del sistema mediante los bloques funcionales.
8. Diseño del código a partir de los bloques.
9. Validación del sistema en co-simulación.

A continuación, se detallarán aquellos puntos de la metodología PowerDigit necesarios para poder diseñar correctamente un lazo de control siguiendo sus pasos, y los puntos en los cuales existen limitaciones y deberán ser abordados por el presente trabajo.

2.1.1. Elección de los parámetros principales del sistema

Tal y como indica el punto 1 del apartado 2.1, en primer lugar se procede a elegir el número de bits del ADC y de la señal PWM. Esta metodología demuestra que no sólo se deberá tener en

cuenta que la resolución del PWM debe ser mayor que la del ADC, sino que también debe tenerse en cuenta la ganancia del sensor para calcular el incremento de tensión del ADC y obtener así una correcta elección del número de bits [1].

A partir del resultado anterior, para calcular el número de bits del PWM no bastará con aplicar la fórmula:

$$\frac{f_{CLK}}{f_{SW}} = nº\ pasos\ contador\ DPWM \quad \textbf{(1)}$$

Donde f_{CLK} es la frecuencia de reloj de la FPGA y f_{SW} la frecuencia de conmutación del convertidor.

Y a partir de la ecuación 1 calcular el número de bits con la relación:

$$2^{n_{bits}} > nºpasos\ contador\ DPWM$$

Donde n_{bits} representa el número de bits.

Sino que será necesario además conocer la topología del convertidor que vamos a utilizar. Esto es así debido a que los convertidores no son sistemas lineales y es necesario linealizar y perturbar para calcular la resolución en tensión de la señal PWM. El diseño de los reguladores desde el punto de vista de la estabilidad se realiza en pequeña señal.

Una vez se establece un valor de resolución de la señal PWM se puede calcular cuántos bits como máximo podremos utilizar para el ADC utilizando la siguiente expresión:

$$\Delta V_{o_{ADC}} = \frac{Rango\ ADC[V]}{2^{nºbitsADC}} \quad \textbf{(2)}$$

Donde $\Delta V_{o_{ADC}}$ es el incremento en tensión medible por el ADC y $Rango_{ADC}$ es el intervalo en el cual el ADC proporciona un valor digital válido.

Siendo $\Delta V'_{o_{ADC}}$ el valor que debe usarse para cumplir con la condición de resolución y que se obtiene de la fórmula 3.

G_{sensor} representa la ganancia introducida por el sensor que acondiciona la salida de tensión o corriente del convertidor a valores medibles por el ADC.

$$\Delta V'_{o_{ADC}} = \frac{\Delta V_{o_{ADC}}}{G_{sensor}} \quad \textbf{(3)}$$

Estos cálculos introducidos formarán la base para definir los parámetros del lazo de control, por ello se han detallado las fórmulas y la base sobre la que se sustentan.

2.1.2. Redondeo en las operaciones aritméticas

Una vez calculados los parámetros principales, se comentan brevemente algunos aspectos relativos al redondeo en las operaciones que se deben tener en cuenta para el diseño del lazo.

La precisión en las operaciones es de gran importancia según la metodología PowerDigit ya que es en esta parte del diseño donde se cometen errores que afectan a la regulación de la planta. Elegir un número de bits inadecuado para las operaciones intermedias y los resultados finales convierte un diseño analítico válido en un incorrecto diseño real. Algunas consideraciones específicas para realizar en el redondeo se citan a continuación:

- Si el bit más significativo de los eliminados es '1', se le debe sumar '1' al resultado obtenido tras la división.

- Si el bit más significativo de los eliminados es '0', no se modifica el resultado obtenido tras la división.

Debido a su complejidad y ya que no es objeto de este trabajo explicar en profundidad la metodología se puede consultar toda la documentación en la referencia bibliográfica [1]. Por otra parte, se han introducido estos conceptos básicos ya que en el capítulo 4 se llevarán a cabo todos los cálculos necesarios para diseñar el sistema de aplicación aquí propuesto, lo cual dará al lector las herramientas suficientes para abordar su propio diseño.

2.1.3. Diseño del sistema completo en bloques funcionales

Una de las principales problemáticas en el diseño de sistemas de control digitales para aplicaciones de potencia es ser capaces de depurarlo a partir de los resultados obtenidos en co-simulación, para el caso de FPGAs, o de resultados experimentales, para el caso de microprocesadores o DSPs. Este problema surge, como ya vimos en el capítulo 1, debido a la falta de un modelo de referencia que sea capaz de reproducir fielmente la implementación hardware llevada a cabo. Para eliminar este problema, la metodología PowerDigit desarrolla un modelo de simulación que tiene en cuenta la implementación hardware; de tal forma que pueden contrastarse los resultados de co-simulación con los de este modelo para detectar, e incluso prever, posibles carencias o fallos en el diseño.

Esta metodología desarrolla un modelo en PSIM tanto de un ADC, como de un regulador y de un modulador digitales. Las principales características de este sistema son:

- Se pueden implementar reguladores clásicos de tipo PID, o los más empleados en sistemas de potencia, como son los denominados tipo 2 y tipo 3.

- Este modelo además permite añadir los retrasos introducidos por la implementación digital y ver qué efectos provocan en la estabilidad del lazo cerrado.

- Por otra parte, este modelo también da la posibilidad al diseñador de implementar lazos de control para modelos de convertidores promediados o conmutados.

Una imagen de la arquitectura que siguen los bloques funcionales puede verse en la figura 2:

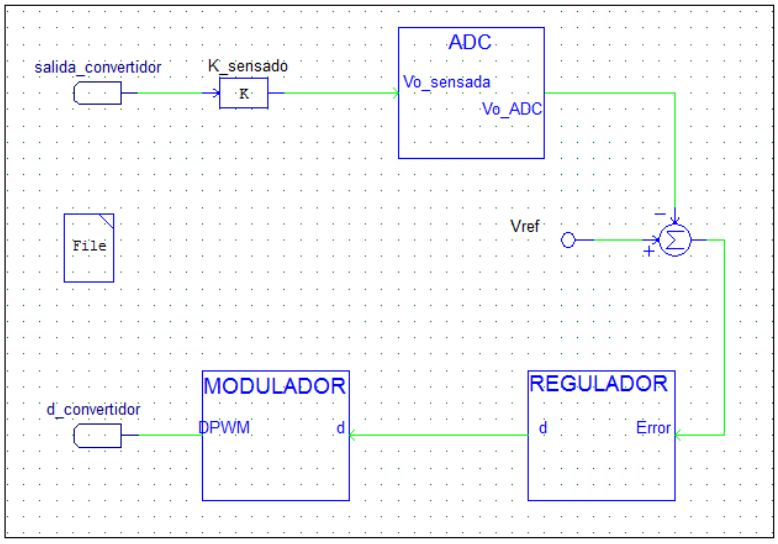


FIGURA 2. SISTEMA DE BLOQUES FUNCIONALES EN PSIM

Vemos que los bloques de la figura 2 se corresponden con la misma estructura descrita en el apartado 1, en el cual se introducía el diagrama típico de control digital para un convertidor.

Otro aspecto a destacar es que este modelo funcional sólo está disponible en PSIM, como ya se comentó en el apartado 1.1. Esto impone una gran limitación para aquel diseñador que no utilice esta herramienta, por lo que el estado actual de la metodología no es independiente de la plataforma donde se desarrolle.

Además, sólo existe un modelo de ADC disponible en bloques funcionales; esto supone que para emplear otro tipo de ADC, como los ADCs con representación en complemento a dos. Sería necesario realizar modificaciones en el código. Lo que dificultaría la labor del diseñador.

Como ya se comentó en el apartado 2.1 no es objeto de este trabajo explicar de forma exhaustiva el desarrollo de la metodología PowerDigit, sino de mostrar un ejemplo de aplicación en todas sus fases de diseño y validación. El lector que desee profundizar más en el desarrollo de dicho modelo puede recurrir nuevamente a la referencia bibliográfica [1] donde podrá encontrar un análisis detallado de cada bloque y de su configuración. Sin embargo, ya que el presente trabajo pretende abordar todas las limitaciones encontradas en PowerDigit, se ha creído conveniente exponer los aspectos más destacados que el lector debe conocer para seguir de forma correcta las fases de diseño.

2.2. Consideraciones al validar el sistema

Tras realizar el diseño de los bloques funcionales se debe diseñar el sistema a implementar mediante lenguaje VHDL. Los bloques servirán como referencia al diseñador a la hora de realizar las simulaciones/co-simulaciones necesarias para validar todo el sistema.

La secuencia de pasos definida para esta validación es la siguiente:

1. Simulación del sistema en bloques funcionales.

2. Diseño del hardware a partir del diseño del punto 1.
3. Co-simulación del sistema para validar el VHDL generado.
4. Comparación de la respuesta entre los bloques funcionales y el VHDL.
5. Depuración del sistema en caso de fallo.
6. Implementación del diseño en la FPGA.
7. Validación experimental del sistema completo.

Si atendemos a los puntos citados, la metodología PowerDigit ha demostrado que es válida hasta el paso 5. Sin embargo, aún no se ha implementado ningún diseño realizado con esta metodología en una FPGA y por lo tanto no se han validado experimentalmente los resultados obtenidos. Es objetivo del presente trabajo cerrar el proceso de diseño hasta el punto 7 y realizar la extensión de la metodología para aplicarla en otro tipo de herramientas de simulación y con otro tipo de elementos, con el fin de eliminar las limitaciones existentes y dotar a la metodología de mayor aplicabilidad.

3. APORTACIONES A LA METODOLOGÍA POWERDIGIT

La metodología PowerDigit ha demostrado que es posible realizar diseños apoyándose en el modelo de bloques funcionales, eliminando así toda la problemática relativa a la depuración del sistema.

Sin embargo, hay limitaciones que es necesario eliminar para dotar de mayor aplicabilidad a la metodología, como ya vimos en los capítulos 1 y 2, y cuyas soluciones serán explicadas en el presente capítulo.

3.1. Diseño e implementación de nuevos bloques funcionales

Una de las principales limitaciones de la metodología PowerDigit es que sólo dispone de un único modelo de ADC para emplear en sus bloques funcionales. Esto implica que es necesario realizar modificaciones en el código VHDL para poder mantener la equivalencia entre modelo de simulación y código hardware.

Si atendemos a la función de transferencia de los ADC, que se entiende como la relación entre el dato digital de salida en relación con la señal analógica de entrada, podemos distinguir dos grandes grupos.

En primer lugar, existen ADCs cuya representación digital es siempre positiva, por lo que adoptan valores en el intervalo $[0 - (2^{n_{bits}} - 1)]$, donde n_{bits} representa el número de bits de salida del ADC. Este tipo de representación es la más común y es la que cubre la metodología PowerDigit.

Sin embargo, otros tipos de ADC tienen una salida que no es siempre positiva, por lo que cuentan con un offset y una ganancia. Este tipo de ADCs requieren por tanto incorporar al diseño en bloques funcionales la posibilidad de configurar la ganancia y el offset con el objetivo de modelar el comportamiento real del conversor, sin necesidad de alterar el código VHDL.

Por tanto, aquí se presenta una solución que pretende solventar esta limitación.

3.1.1. Bloque ADC

Como ya se ha comentado en la introducción del capítulo, una de las principales limitaciones de PowerDigit es que no cuenta con varios modelos de ADC. Por ello, el nuevo modelo incorporado a la metodología es el que se muestra en la figura 3:

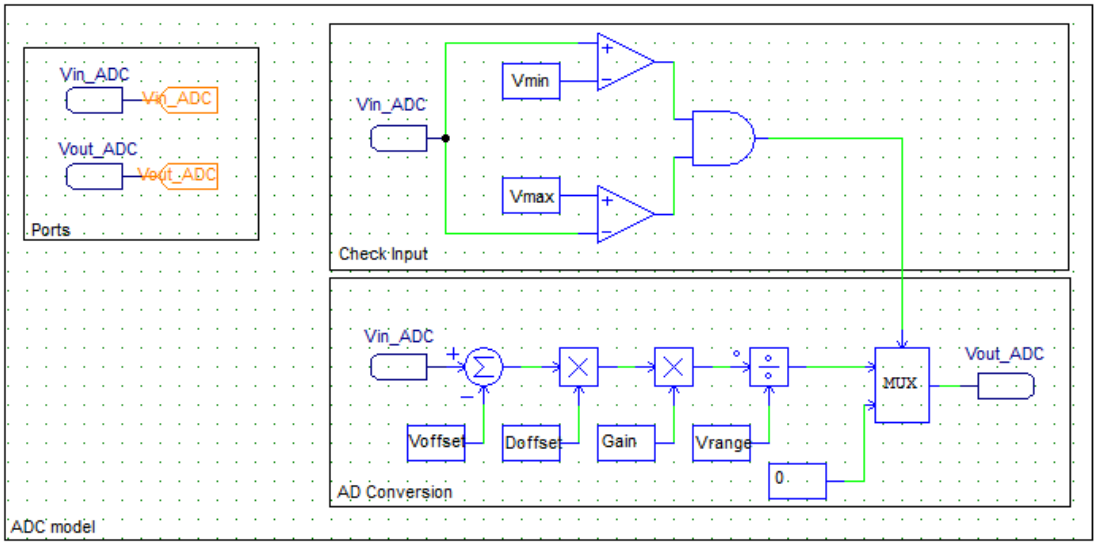


FIGURA 3. NUEVO MODELO DE CONVERSOR ANALÓGICO-DIGITAL

Se ha dividido el modelo en varios bloques para facilitar su explicación y hacer que el lector comprenda más fácilmente su estructura.

- **Ports**. Son los puertos de entrada y salida que conectan el modelo de ADC con el resto del sistema.
- **Check Input**. Esta parte del modelo es necesaria para evitar que cualquier valor de tensión fuera del rango admisible permita obtener un valor digital a la salida. Por tanto, sirve como parte de control para el bloque ADC model.
- **AD Converter**. Esta parte representa la ecuación general que modela los ADC con offset y ganancia.

El modelo dispone de parámetros configurables para modificarlos en función del ADC que se utilice en cada momento. El detalle de cada parámetro se explica a continuación:

- ***Vmin*** y ***Vmax*** representan los valores mínimo y máximo respectivamente que es capaz de leer el ADC por su entrada analógica.
- ***Voffset*** representa el valor de tensión central del ADC, es decir, el valor de tensión que se corresponde con el valor digital cero. Es obvio pensar que para el caso de un ADC convencional este parámetro es cero, tal y como se comentó en el apartado 3.1.
- ***Doffset*** representa el valor digital de escalado. Este parámetro se calcula con la expresión $2^{n_{bitsADC}}$ para el caso de un ADC convencional, y $2^{n_{bitsADC}-1}$ para el caso de un ADC con representación de la salida en complemento a dos (valores negativos y positivos).
- ***Gain*** permite la posibilidad de añadir una ganancia adicional en función del tipo de ADC. En algunos casos existen bloques amplificadores que modifican la ganancia del ADC y deben ser tenidos en cuenta. Si no es necesaria ninguna ganancia adicional se configurará como 1.
- ***Vrange*** representa el valor de tensión entre el cual puede desplazarse el valor de offset de la entrada para poder leer a la salida un dato.

Llegados a este punto, un diseñador que utilice la metodología PowerDigit puede elegir entre implementar este nuevo bloque o el que ya existía anteriormente en función del ADC que utilice.

Para ello sólo será necesario sustituir en la figura 2 del apartado 2.1.3 el bloque ADC por este nuevo bloque.

3.2. Adaptación de la metodología empleando MATLAB

En la actualidad, existen numerosas herramientas de modelado y simulación de sistemas que están en mayor o menor medida incorporadas a las empresas del sector. Sin duda, MATLAB es probablemente la más popular de todas ellas. Además, incorpora una infinidad de herramientas adicionales que dan mayor potencia a los diseños realizados en esta plataforma.

Por otro lado, como ya vimos, imponer de antemano un software específico a un diseñador le resta flexibilidad a la metodología; esta limitación supone la necesidad de desarrollar nuevos modelos en bloques funcionales que permitan demostrar la independencia de la metodología respecto del software de simulación que se utilice.

Debido a esto, aquí se propone un diseño en bloques funcionales equivalente al modelo en PSIM desarrollado en MATLAB que permita extender la metodología de bloques funcionales a otras plataformas de desarrollo.

3.2.1. Desarrollo detallado del modelo en MATLAB

El objetivo que se pretende alcanzar es demostrar la equivalencia de los modelos en bloques funcionales independientemente de la herramienta donde se desarrollen. Por ello se ha partido como base del modelo de PSIM para realizar el nuevo modelo en MATLAB. Toda la información necesaria sobre el modelo PSIM se encuentra en la referencia [1] de la bibliografía al igual que ya se citó en el apartado 2.1.3.

El diagrama de bloques de más alto nivel del lazo es el siguiente:

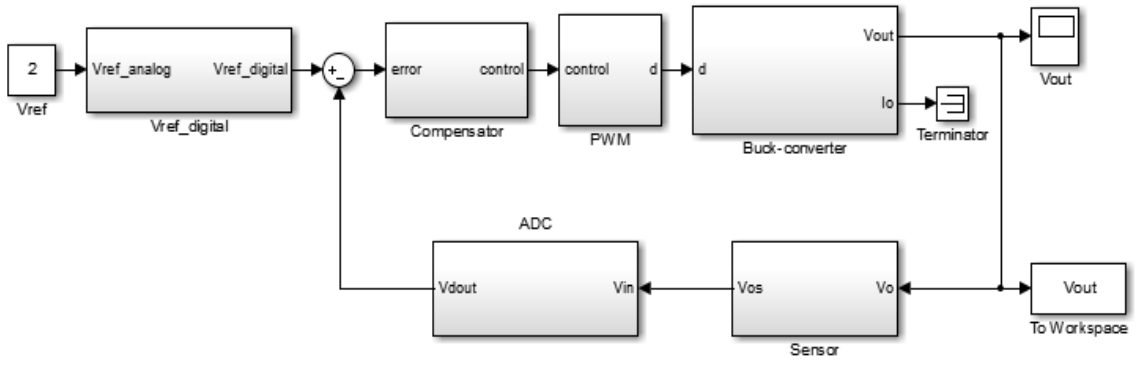


FIGURA 4. DIAGRAMA DE BLOQUES MODELO FUNCIONAL LAZO DE CONTROL MATLAB

Como podemos apreciar en la figura 4, la estructura de los bloques es similar al de cualquier otro sistema realimentado al igual que vimos en la figura 1 de la introducción. Los bloques que constituyen el sistema son:

- Convertidor reductor. Será la planta a controlar por nuestro lazo de control.

- Sensor de tensión. Acondiciona la señal de tensión a un rango medible por el ADC.
- Conversor analógico/digital. Representa el modelo del ADC que se empleará experimentalmente.
- Regulador. Es el bloque donde se realizan los cálculos de control.
- Modulador. Genera la señal PWM que dispara el mosfet del convertidor.
- Referencia. Este bloque devuelve el valor de tensión de referencia que requiere el sistema para su funcionamiento.

Junto con el modelo de bloques funcionales de MATLAB se desarrolla un pequeño script con todos los valores de los parámetros necesarios para que funcione la simulación. Esto sería equivalente a los ficheros file que utiliza PSIM y que pueden consultarse en [1].

Sin embargo con este fichero *.m* está todo más centralizado, ya que PSIM asocia a cada subcircuito un fichero independiente en formato *.txt*.

A continuación se va a proceder al estudio detallado de cada bloque:

- **<u>Convertidor reductor</u>**

Pese a que la planta no pertenece directamente a la metodología de bloques funcionales, cabe destacar que se ha realizado utilizando la toolbox de MATLAB SimPowerSystem.

Esta herramienta permite simular de forma sencilla circuitos electrónicos pudiendo añadir todo tipo de efectos reales, como parásitos, e incluso puede conectarse al resto de componentes de Simulink para un post-procesado de los resultados.

En el capítulo 4 se hablará en detalle de las características del convertidor utilizado para este trabajo.

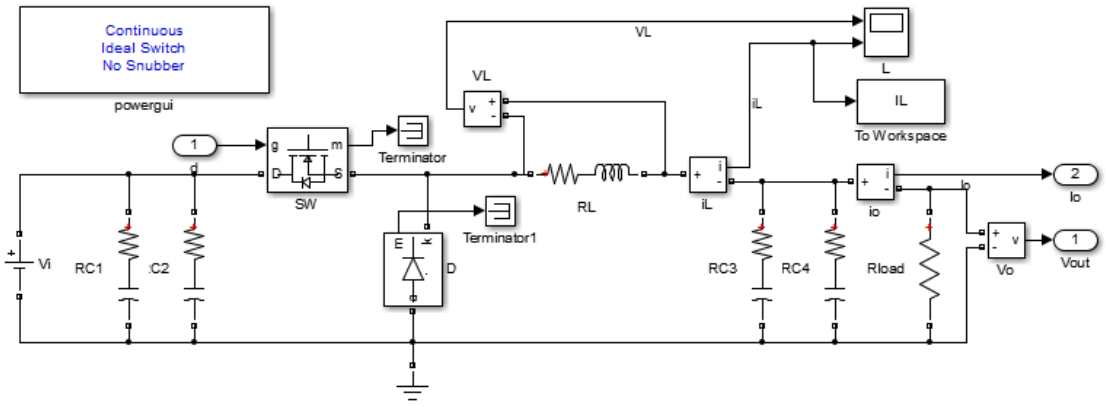


FIGURA 5. CONVERTIDOR REDUCTOR USANDO LA TOOLBOX SIMPOWERSYSTEM

- **Sensor de tensión**

El sensor de tensión sirve para adaptar la tensión de salida del convertidor a un valor medible por el ADC. En nuestro caso se coloca una ganancia que modela el comportamiento de un divisor resistivo clásico.

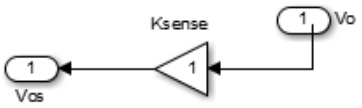


FIGURA 6. SENSOR DE TENSIÓN

- **Convertidor analógico/digital**

El ADC obtiene un valor analógico de tensión proveniente del sensor y lo convierte a un valor entero, que representa el valor digital equivalente que le llega a la FPGA. Este modelo de ADC es básicamente una ganancia a la que se le incorpora un retraso debido a la conversión del dato. La única peculiaridad es la incorporación de un bloque de *Sample and Hold* controlado por un comparador. Este sistema pretende modelar las prestaciones de algunos ADCs que tienen su origen de medida descentrado de cero. Por ejemplo, en el caso de un ADC de ventana con un rango entre [0,5-5] V. El ADC no dará valores nuevos a la salida hasta que el valor muestreado supere los 0,5V. Por ello, la utilización del offset y del comparador es necesaria de cara a dar mayor flexibilidad al diseño y poder utilizar distintos tipos de ADC usando esta misma arquitectura.

Este modelo de ADC aquí desarrollado permite unificar los dos modelos de PSIM que se comentaron en el apartado 3.1.1.

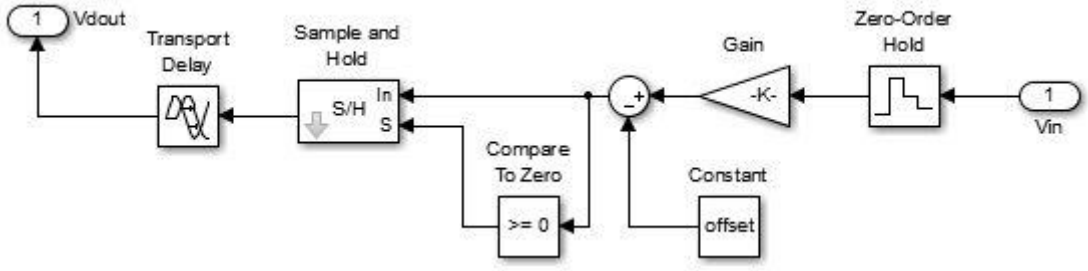


FIGURA 7. MODELO DEL ADC DE VENTANA

La ganancia del ADC se modela en la siguiente ecuación:

$$K_{ADC} = \frac{2^{nbitsADC}}{V_{adcMAX} - V_{adcMIN}} \quad (4)$$

Donde V_{adcMAX} y V_{adcMIN} representan los valores máximo y mínimo de tensión medible por el ADC respectivamente.

Los bloques de ganancia y retraso tienen parámetros que serán explicados en el capítulo 4. Por lo que no es necesario detallar su obtención en este apartado.

A diferencia del modelo inicial de PSIM, este modelo requiere del parámetro *offset* para utilizar el bloque ADC. La obtención de este parámetro se realiza de la siguiente forma:

$$Offset = K_{ADC} \cdot V_{adcMIN} \quad (5)$$

El offset se utiliza para que el valor mínimo de tensión V_{adcMIN} corresponda con una salida cero. Es lógico pensar que para $V_{adcMIN} = 0$, el offset será cero. Pero para $V_{adcMIN} \neq 0$, el valor de offset no es cero, y es necesario calcularlo. Por ello se plantea la ecuación anterior.

Para el caso de utilizar ADCs como el propuesto en el apartado 3.1.1, el valor de offset debe mantenerse en cero, puesto que en ese caso el ADC si puede entregar valores digitales menores que cero. Por lo que el uso de la ecuación 5 está condicionado por el tipo de ADC a emplear.

- **Regulador**

Este bloque es el encargado de realizar todas las operaciones aritméticas necesarias para la obtención de ciclo de trabajo que controla el convertidor. Su diseño es el más complejo ya que está compuesto de gran cantidad de elementos.

Pese a que se ha intentado mantener la misma estructura que en el modelo de PSIM no existen equivalencias exactas entre los bloques de un simulador y de otro, por lo que han sido necesarias ciertas modificaciones en algunos elementos. El bloque regulador además cuenta con un filtro AntiWindup para evitar que el integrador siga funcionando una vez hemos superado los límites de ciclo de trabajo superior o inferior. Una explicación detallada de cada parte del bloque regulador se encuentra en la bibliografía [1], en este trabajo se explicarán con más detalle aquellos elementos que han sido modificados de un modelo frente a otro.

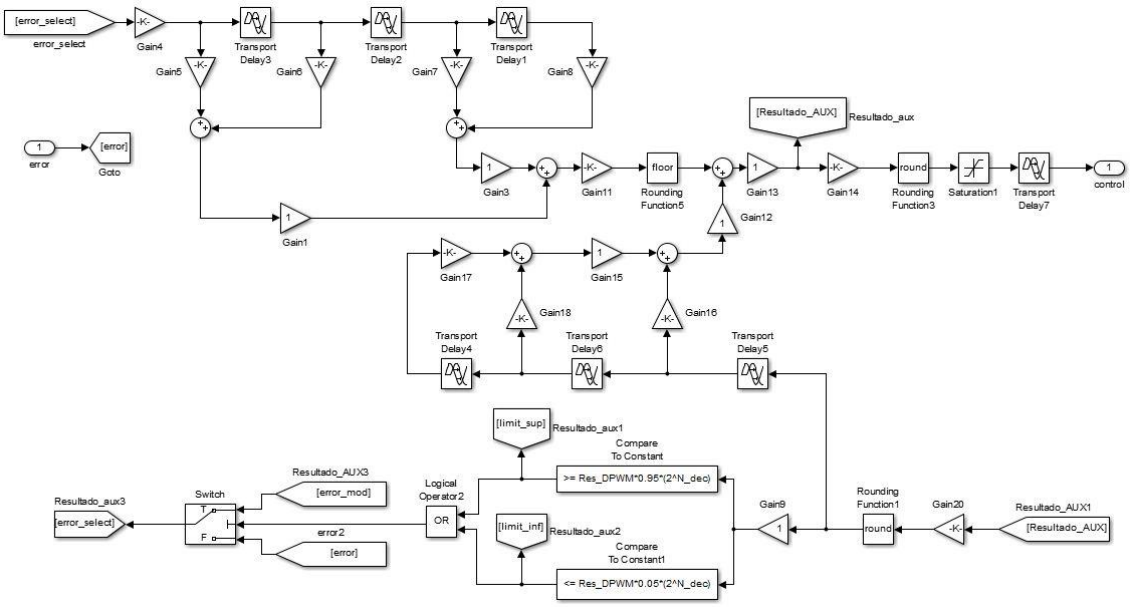


FIGURA 8. ARQUITECTURA DEL REGULADOR DIGITAL

Hay situados un par de elementos *Compare To Constant* que sirven para obtener un valor lógico en función del valor del error que recibe. Este bloque sustituye los operacionales en modo comparador que están implementados en PSIM.

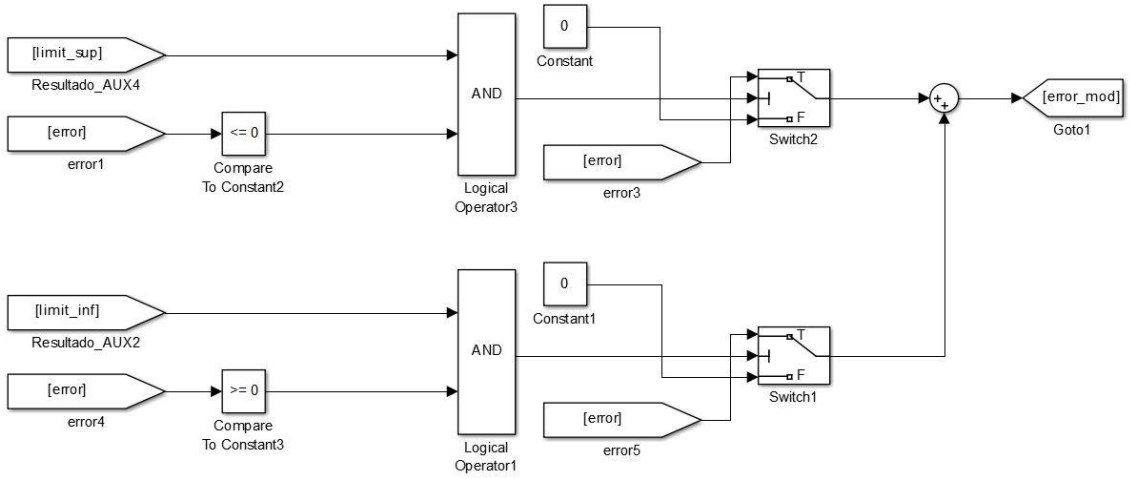


FIGURA 9. CÁLCULO ERROR MODIFICADO DEL FILTRO ANTIWINDUP

Aparece también un elemento *Transport Delay* que representa el retraso debido a la conversión del dato proveniente del ADC. Se utiliza para evitar que la señal de muestreo empiece a generarse al inicio de la simulación, sino cuando el ADC ya tenga un dato válido. Además, el bloque *Sample and Hold* de PSIM no es necesario incorporarlo ya que la señal "*Resultado_AUX*" se muestrea cada ciclo de conmutación, lo que evita que tengamos varias correcciones de ciclo de trabajo por período. Una comparativa detallada de los bloques PSIM/MATLAB puede verse en la tabla 1 al final de este apartado.

- **Modulador**

El bloque modulador es el encargado de generar la señal PWM que servirá para disparar el/los dispositivo/s de control del convertidor. El bloque recibe una señal de control del bloque regulador con el nuevo valor de ciclo de trabajo y el modulador lo compara con una señal triangular para generar dicha señal PWM. El modelo de modulador es el que se muestra en la figura siguiente.

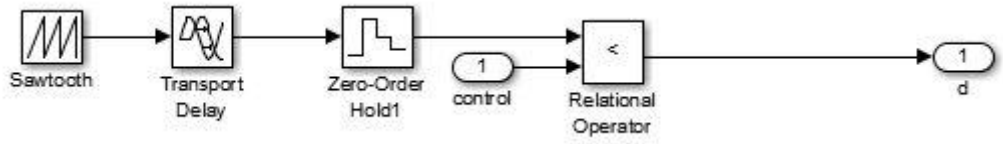


FIGURA 10. BLOQUE MODULADOR

- **Referencia**

El bloque de referencia es similar al de ADC excepto porque no tiene ningún retraso a la salida. Esto es así porque el valor de la referencia está directamente programado en la FPGA como una constante. Para configurar de forma sencilla un nuevo valor en el simulador se utiliza esta estructura, que devuelve el valor entero digital equivalente al valor de tensión analógico de entrada.

Otra diferencia respecto al modelo de ADC es que no cuenta con el muestreador y el comparador. Esto es así debido a que el bloque de referencia puede poner a su salida cualquier valor de tensión, por lo que implementar esa funcionalidad en este bloque no sería correcto.

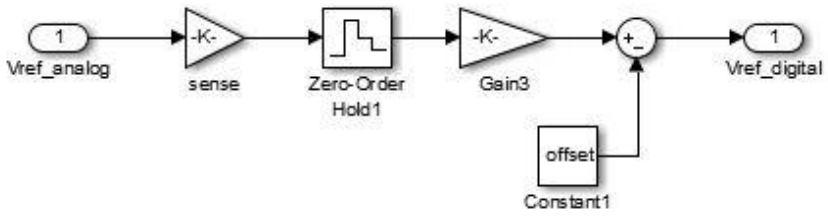


FIGURA 11. BLOQUE GENERADOR DE REFERENCIA

En la tabla siguiente puede verse un resumen de la equivalencia de bloques utilizada para la implementación mediante bloques funcionales usando PSIM y MATLAB.

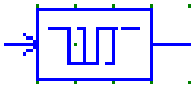
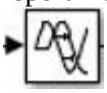
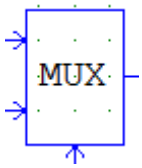
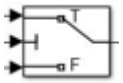
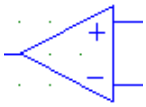
BLOQUE PSIM	BLOQUE MATLAB
Delay 	Transport Delay 
Sample and Hold 	*Este bloque no ha sido necesario en MATLAB ya que es posible definir el tiempo de muestreo de cada bloque, a diferencia de lo que ocurre en PSIM.*
Mux 	*Switch* 
Comparator 	*Compare To Constant**  **puede ser cero u otro valor*

TABLA 1. EQUIVALENCIA ENTRE BLOQUES PSIM Y MATLAB

Tras realizar el modelo en MATLAB y para garantizar que son equivalentes se deben validar ambos sistemas en simulación. Con el objetivo de mantener la estructura descrita en el apartado 1.4, todas las validaciones se llevarán a cabo en el capítulo 5. Por lo que el lector que desee comprobar los resultados obtenidos deberá acudir a dicho capítulo, apartado 5.1.1.

3.3. Modificación de la descripción del circuito digital en VHDL

En este apartado se explican detalladamente todas las modificaciones realizadas en el código con el objetivo de mejorar la metodología, buscando siempre dotarla de mayor aplicabilidad; al igual que en los apartados 3.1 y 3.2.

3.3.1. Aumento de la resolución del sistema

En los sistemas de control digitales es necesario implementar un modulador que genere la señal de control PWM para disparar los dispositivos de la etapa de potencia. Este bloque necesita de gran precisión para evitar problemas de inestabilidad, por lo que frecuentemente requiere trabajar a una velocidad superior a la del resto del circuito digital. Hasta el momento, la metodología PowerDigit no contempla esta situación, lo cual limita bastante qué tipo de diseños son posibles de realizar usando esta metodología. En el caso de frecuencias de conmutación pequeñas, del orden de menos de 100kHz, el reloj de cualquier FPGA, en torno a las decenas o centenas de MHz, es suficiente para generar el PWM con cierta precisión. Sin embargo, cuando las frecuencias de conmutación aumentan, es necesario incorporar un bloque que aumente el reloj del bloque modulador, y por consiguiente, la resolución del PWM.

Para llevar a cabo este aumento de la frecuencia de reloj se utiliza un bloque IP para el control de señales de reloj, que permita multiplicar la frecuencia de la señal de reloj utilizada como referencia.

Un esquema del diseño propuesto puede verse en la figura siguiente:

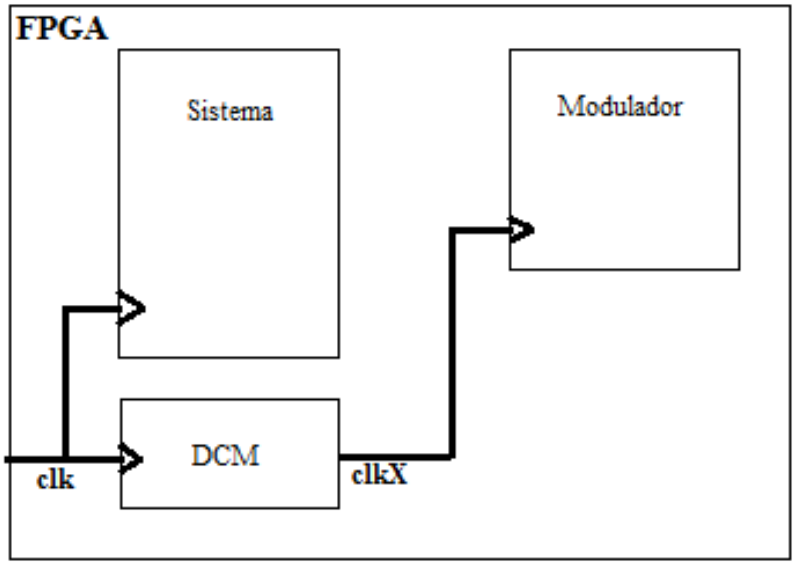


FIGURA 12. ESQUEMA CONEXIÓN RELOJES DEL SISTEMA

Vemos como en este caso, el DCM es el encargado de generar la señal de reloj del bloque modulador, y a través de unos parámetros configurables aumentar la frecuencia del bloque, con el consiguiente aumento de la resolución del PWM.

Una vez desarrollada esta modificación, se elimina la limitación de no poder diseñar sistemas aplicando la metodología PowerDigit que requieran de frecuencias de conmutación muy elevadas.

3.3.2. Diseño genérico

Un aspecto a tener en cuenta cuando se realizan diseños a cualquier nivel es facilitar la reconfiguración de los parámetros para adaptarlos rápidamente a nuevos diseños. En el caso del diseño VHDL el uso de genéricos permite reconfigurar el código fácilmente modificando sus valores por defecto, por lo que readaptar un diseño se convierte en una tarea muy simple.

Pese a que la metodología PowerDigit ya contaba con algunos genéricos, estos no abarcaban la totalidad de los parámetros a configurar por el diseñador, lo que obligaba a modificar manualmente algunas señales y constantes del propio diseño.

Este motivo ha llevado a modificar la estructura de las entidades con el objetivo de poder reconfigurar el diseño teniendo que acceder únicamente a un solo fichero. Los nuevos bloques incorporados a la metodología se han desarrollado manteniendo la estructura de genéricos que aquí se propone.

Con el fin de centralizar todos los parámetros en un único lugar se declaran todos ellos en la entidad más alto nivel del diseño, se dará visibilidad a los genéricos en los bloques que corresponda y todos los bloques heredarán del fichero top.

Por lo tanto, si se cambia un genérico del fichero top, se verá su cambio en los bloques donde corresponda. Además, siguiendo esta metodología, incorporar nuevos genéricos es muy sencillo, facilitando posibles tareas de mejora de la PowerDigit.

3.4. Generación automática de código VHDL a partir de plantillas

La estructura de genéricos comentada en el apartado 3.3.2, como ya vimos, permite configurar fácilmente los parámetros del sistema. El diseñador puede acceder al fichero de alto nivel y modificar manualmente los genéricos para usarlo en su diseño particular.

Sin embargo, ya ha introducido gran cantidad de estos parámetros en los ficheros FILE de PSIM para realizar la validación de su sistema en bloques funcionales. Por lo que tener que acceder nuevamente a “re-configurar” los parámetros en el VHDL sería realizar dos veces la misma acción.

Puesto que los bloques funcionales surgen como paso previo al diseño hardware, todos los parámetros de PSIM son exportables al diseño VHDL; por ello, se ha desarrollado un programa en C que almacena los valores de los FILE de PSIM y los exporta al diseño de alto nivel de VHDL a partir de una plantilla con valores por defecto.

El diagrama de flujo que describe la secuencia del código es la siguiente:

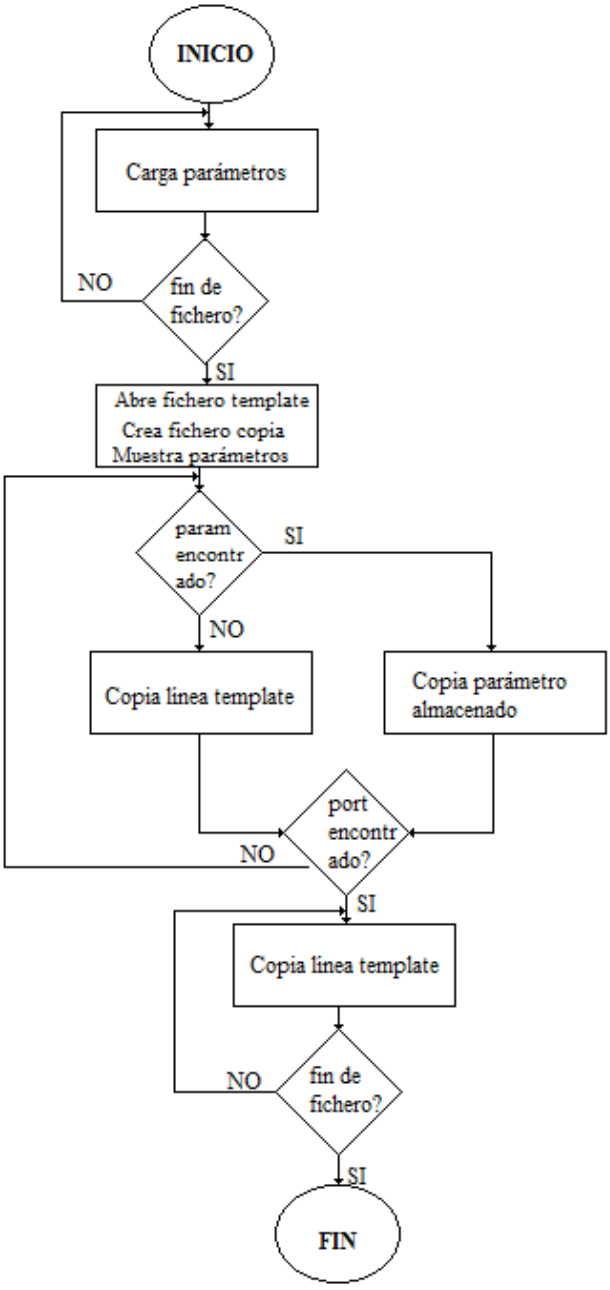


FIGURA 13. DIAGRAMA DE FLUJO DE PROGRAMA DE GENERACIÓN AUTOMÁTICA

A continuación se hace un repaso más detallado de la secuencia que sigue el código:

1. El programa abre los ficheros FILE y carga los parámetros que se encuentra en una estructura.
2. Una vez se alcanza el fin del fichero lo cierra y muestra por pantalla todos los parámetros almacenados. Posteriormente abre el fichero template y crea un fichero copia en el directorio correspondiente.
3. El programa busca línea a línea los parámetros que tiene almacenados, en caso de encontrar uno, lo copia en el fichero creado, de lo contrario copia la línea de template y salta a la línea siguiente.
4. Cuando el programa encuentra la palabra clave “Port” finaliza la búsqueda. A partir de este punto copia el resto del fichero template en el fichero creado.

- Un ejemplo de funcionamiento se muestra a continuación:

El usuario dispondrá de un fichero ".exe" que contendrá la información del programa. Una vez lo ejecute le aparecerá la pantalla que indica la figura 14:

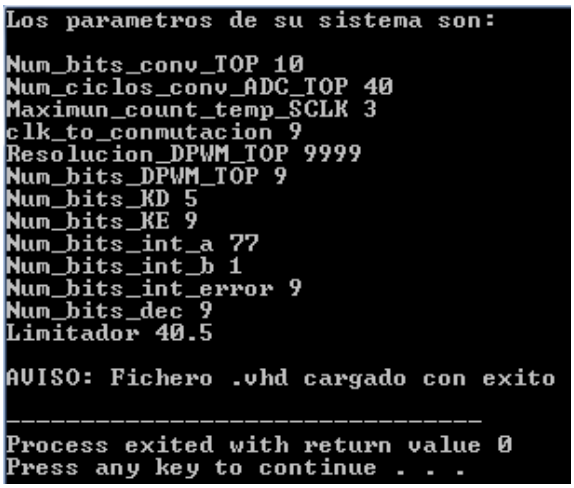


FIGURA 14. EJEMPLO DE APLICACIÓN CÓDIGO AUTOMÁTICO

En la figura se muestran todos los parámetros encontrados por el programa y su valor correspondiente. Una vez ejecutado, el usuario encontrará en la carpeta de diseño un fichero ".vhd" generado a partir de la plantilla disponible con los nuevos parámetros ya configurados, en este caso el fichero generado contiene:

```
entity Lazo_de_Control is
     Generic (
                 Num_bits_conv_TOP   : integer := 10 ;
                 Num_ciclos_conv_ADC_TOP   : integer := 40 ;
                 Maximun_count_temp_SCLK   : integer := 3 ;
                 clk_to_conmutacion   : integer := 9 ;
                 Resolucion_DPWM_TOP   : integer := 9999 ;
                 Num_bits_DPWM_TOP   : integer := 9 ;
                 Num_bits_KD   : integer := 5 ;
                 Num_bits_KE   : integer := 9 ;
                 Num_bits_int_a   : integer := 77 ;
                 Num_bits_int_b   : integer := 1 ;
                 Num_bits_int_error   : integer := 9 ;
                 Num_bits_dec   : integer := 9 ;
                 Limitador   : integer := 40.5 );
    Port ( CLK : in  STD_LOGIC;
           RESET : in  STD_LOGIC;
           SDATA : in  STD_LOGIC;
           CS : out  STD_LOGIC;
           SCLK : out  STD_LOGIC;
           DPWM : out  STD_LOGIC);
end Lazo_de_Control;
```

FIGURA 15. CONFIGURACIÓN DE LOS GENÉRICOS DE LA ENTIDAD TOP

Si nos fijamos detenidamente, vemos como los valores de los parámetros de la figura 15 coindicen con los valores almacenados por el programa. A partir de la palabra "Port" el fichero permanece exactamente igual que el template.

Tras el ejemplo mostrado anteriormente, en el que puede verse la configuración del código de forma automática, se cierra la fase de automatización del diseño. Por lo tanto, un diseñador que desee configurar el diseño hardware para su sistema no requerirá conocimientos de VHDL puesto que a través del programa desarrollado será capaz de configurarlo sin necesidad de abrir ningún fichero "*.vhd*".

4. CASO DE ESTUDIO

4.1. Características de los elementos del sistema

Puesto que el objetivo que persigue este trabajo es validar experimentalmente la metodología PowerDigit, en este apartado se van a detallar las características del convertidor y de la FPGA que se van a emplear para las pruebas.

4.1.1. Características del convertidor DC/DC

El convertidor elegido es el modelo PTD08A010W del fabricante Texas Instrument. Es un convertidor de topología reductora, no aislado, y cuyas características principales se exponen en la tabla 2. Además de las características eléctricas del convertidor, el datasheet proporciona información acerca del montaje recomendado para una aplicación estándar tal y como se muestra en la figura 17.

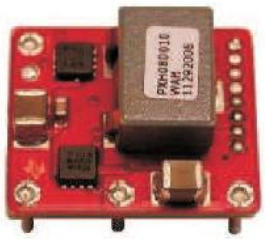


FIGURA 16. CONVERTIDOR PTD08A010W [13]

Tensión entrada [V]	4,75 – 14	**C_{i1} [µF]**	330
Tensión salida [V]	0,7 – 3,6	**C_{i2} [µF]**	22
Frecuencia [kHz]	300 – 1000	**C_{O1} [µF]**	47
Corriente salida [A]	10	**C_{O2} [µF]**	330
Bobina [µH]	0,9	**ESR [µF]**	1,5
DCR [mΩ]	2,2	**Potencia salida [W]**	10

TABLA 2.CARACTERÍSTICAS ELÉCTRICAS DEL CONVERTIDOR

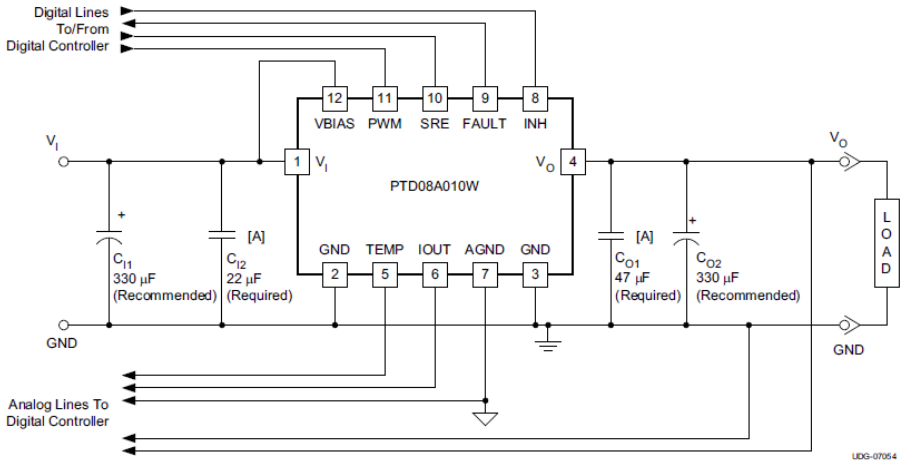


FIGURA 17. MONTAJE DETALLADO PARA APLICACIÓN ESTÁNDAR DEL PTD08A010W [13]

Para facilitar las pruebas que posteriormente se realizarán, el departamento facilita una PCB donde está montado el convertidor con el resto de componentes que se muestran en la figura anterior.

4.1.2. Características de la placa FPGA

Una FPGA es un dispositivo que contiene bloques de lógica configurables mediante diversos lenguajes específicos, en este caso, VHDL. La mayoría de las FPGAs son reprogramables, lo que proporciona al diseñador una gran flexibilidad a la hora de realizar modificaciones. Sin embargo, son más lentas y consumen mayor potencia que otros dispositivos de aplicación específica disponibles en el mercado. Para este proyecto la mejor opción es utilizar una FPGA debido a que utilizar circuitos de aplicación específica sería mucho más costoso.

La FPGA elegida para realizar el trabajo es la SPARTAN 3E del fabricante XILINX. Es una placa sencilla y de bajo coste, disponible en la universidad, con un oscilador interno de 50MHz y que incorpora algunos periféricos interesantes. Una imagen general de la FPGA puede verse en la figura 18.

Además de su bajo coste y su disponibilidad, otro aspecto a destacar es que entre sus periféricos cuenta con un ADC cuyo rango medible está entre [0,4 – 2,9] V y con obtención de datos digitales negativos. Como ya vimos en los apartados anteriores, la metodología PowerDigit no cubre la posibilidad de utilizar ADCs de este tipo y por tanto realizar las validaciones con este convertidor permitiría extender la metodología.

FIGURA 18. PLACA FPGA SPARTAN 3E [12]

En la figura 19 pueden verse en detalle los elementos que componen el ADC. Los pines VINA y VINB permiten introducir señales de tensión analógicas, los pines REF y VCC sirven para comprobar los valores de referencia internos y el valor de alimentación del bloque respectivamente y el pin GND sirve como punto de conexión a masa.

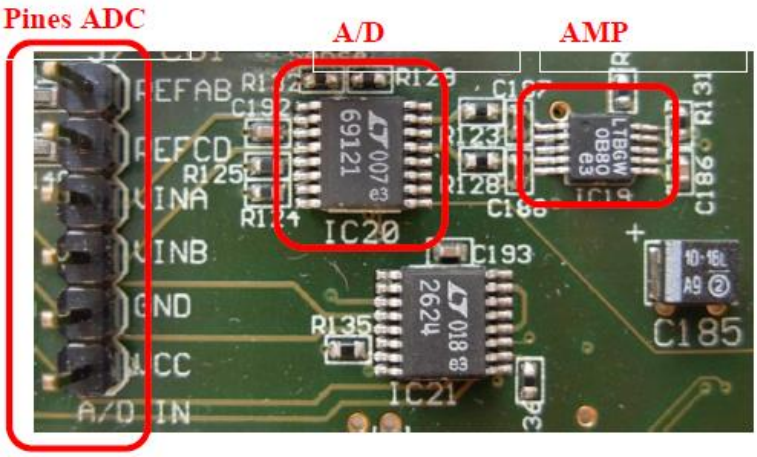


FIGURA 19. CONVERSOR ANALÓGICO-DIGITAL DE LA FPGA [11]

4.1.2.1. Características del circuito de captura analógica

El circuito de captura analógica dispone de dos canales; además cuenta con un pre-amplificador programable y un conversor analógico-digital, ambos controlados por la FPGA. El diagrama de conexión entre los distintos elementos puede verse en la figura 20.

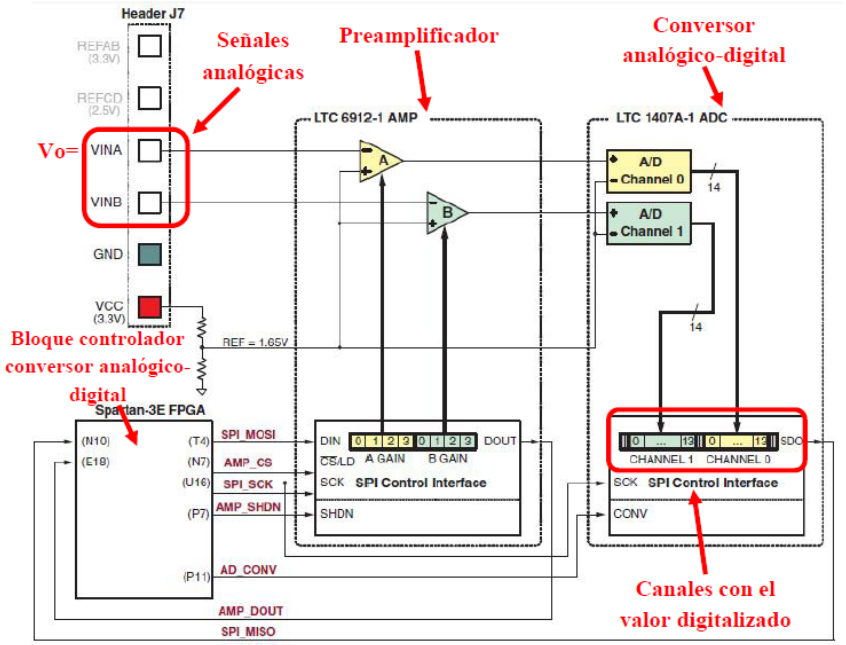


FIGURA 20. VISTA DETALLADA DEL CIRCUITO DE CAPTURA ANALÓGICA [11]

El resultado de la conversión de las señales analógicas VINA/B es un vector de 14 bits en complemento a dos para cada una de las señales, asignando el canal 0 a VINA y el canal 1 a VINB. La ecuación que expresa esta conversión es la siguiente [12]:

$$D[13:0] = GAIN \cdot \frac{V_{IN} - 1{,}65\ V}{1{,}25\ V} \cdot 8192 \qquad (6)$$

Donde GAIN representa la ganancia programada en el pre-amplificador y V_{IN} la señal de entrada analógica.

El rango máximo del ADC es ±1,25 V, centrado alrededor del voltaje de referencia, 1,65 V. Toda la información opcional que el lector desee puede encontrarla en la hoja de características de la FPGA que se encuentra en la bibliografía [12].

4.1.2.1.1. Características del pre-amplificador

El pre-amplificador cuenta con dos amplificadores independientes de ganancia programable. El objetivo de este bloque es ajustar los valores medibles a la entrada del ADC en función de la ganancia programada. La tabla 3 agrupa las posibles combinaciones de ganancia y el rango que admite el ADC bajo esas condiciones.

Ganancia	A3	A2	A1	A0	Rango voltaje entrada	
	B3	B2	B1	B0	Mínimo	Máximo
0	0	0	0	0		
-1	0	0	0	1	0,4	2,9
-2	0	0	1	0	1,025	2,275
-5	0	0	1	1	1,4	1,9
-10	0	1	0	0	1,525	1,775
-20	0	1	0	1	1,5875	1,7125
-50	0	1	1	0	1,625	1,675
-100	0	1	1	1	1,6375	1,6625

TABLA 3. CONFIGURACIONES DE GANANCIA BLOQUE PRE-AMPLIFICADOR

Este bloque utiliza un protocolo de comunicaciones serie SPI, por lo que un único bit es almacenado en cada ciclo de reloj. Las figuras 21 y 22 muestran información acerca de la interfaz de conexión entre pre-amplificador/ FPGA y el timing durante la comunicación entre bloques.

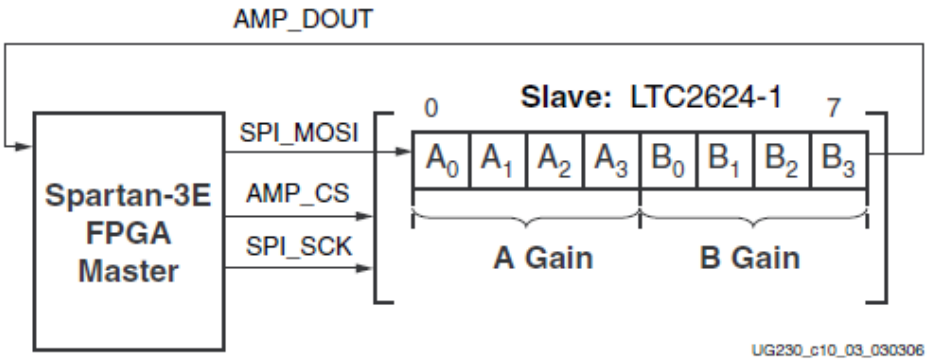


FIGURA 21. INTERFAZ SPI/AMPLIFICADOR [12]

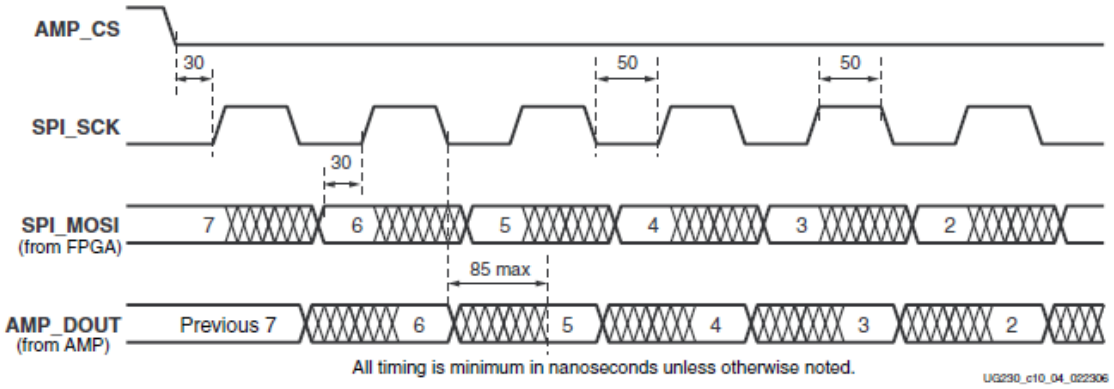


FIGURA 22. CRONOGRAMA DE COMUNICACIÓN ENTRE FPGA/AMPLIFICADOR [12]

Si el lector desea profundizar más acerca del funcionamiento del bloque puede recurrir a la hoja de características de la FPGA [12], donde puede encontrarse toda esta información de forma detallada.

4.1.2.1.2. Características del conversor analógico-digital

El conversor analógico-digital cuenta con dos canales de entrada analógicos, como ya vimos en el apartado 4.1.2.1, que son muestreados de forma simultánea. Al igual que el bloque pre-amplificador, el protocolo de comunicación que utiliza es SPI.

La figura 23 muestra la interfaz de conexión entre el conversor y la FPGA además de un diagrama en la cual se representa la carga de un dato completo en ambos canales.

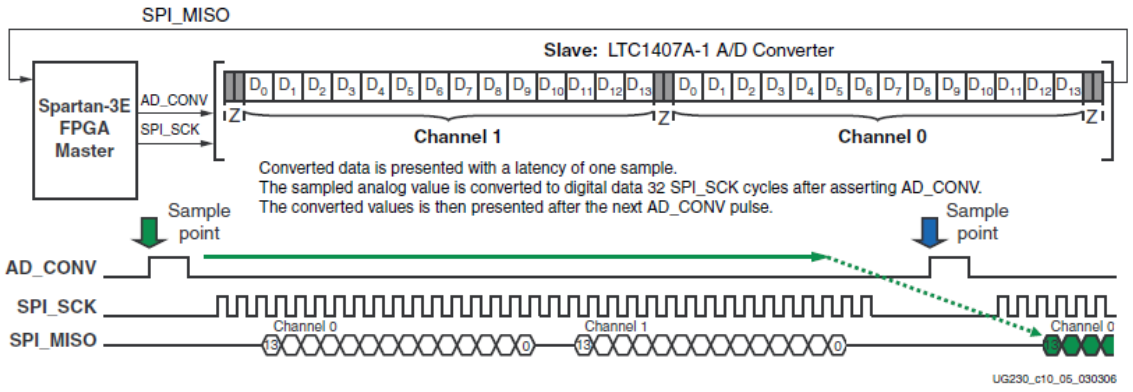


FIGURA 23. INTERFAZ ENTRE CONVERSOR Y FPGA [12]

Como puede leerse en la propia figura, el dato muestreado en el instante actual se mostrará con un retraso de un ciclo de muestreo. Por lo que el dato muestreado en el instante que indica la flecha verde se cargará al final del ciclo que comienza con el muestreo de la flecha azul.

Por otra parte, en la figura se aprecia cómo tras iniciar el muestreo se cargan los 14 bits del canal 0 y después los 14 bits del canal 1, guardando unos ciclos en blanco para asegurar que la conversión se realiza de forma adecuada.

Otro aspecto importante es qué limitación nos impone el sistema a la hora de elegir la frecuencia del reloj, la señal SPI_SCK. Si atendemos a la información proporcionada por el datasheet, ver figura 24, y haciendo un simple cálculo obtenemos:

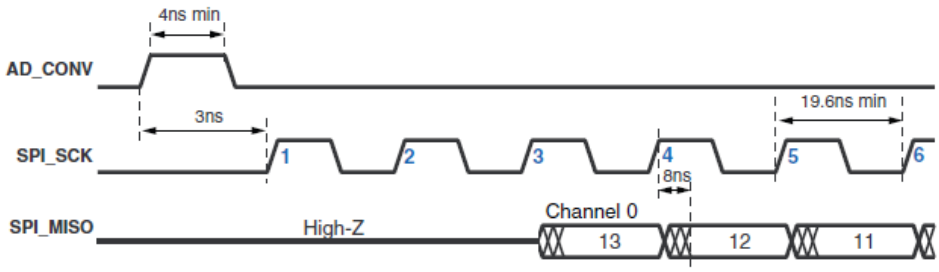


FIGURA 24. DETALLE DE LA TRANSFERENCIA DE DATOS DEL ADC [12]

$$f_{SPI\_SCK\_MAX} = \frac{1}{T_{min}} = \frac{1}{19.6\ ns} = 51MHz \quad (7)$$

Con lo cual, la frecuencia del reloj SPI no podrá exceder los 51MHz.

4.1.2.1.3. Desactivación de señales

Según el datasheet hay señales que son compartidas por otros periféricos que están en la placa por lo que para que todo funcione correctamente es necesario desactivar estos dispositivos cuando se haga uso del amplificador o del ADC. La tabla 4 está extraída del datasheet [12] y hace referencia a las señales que es necesario desactivar:

Nombre señal	Dispositivo desactivado	Valor
SPI_SS_B	SPI Serial Flash	1
DAC_CS	DAC	1
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	0
AMP_SHDN	Active-High shutdown, reset	0

TABLA 4. DESACTIVACIÓN DE SEÑALES

4.2. Especificaciones del sistema

Una vez conocidas las características del convertidor y de la FPGA se van a detallar las especificaciones impuestas que debe cumplir el diseño así como los cálculos necesarios para la obtención de todos los parámetros que cumplan con los requisitos.

El convertidor deberá funcionar en el punto de trabajo que establece la tabla 5.

Tensión de entrada [V]	**Tensión de salida [V]**	**Frecuencia [kHz]**
6	2	400

TABLA 5. ESPECIFICACIONES DEL SISTEMA

A partir de la tabla anterior es inmediato obtener que el valor de la ganancia programable para el amplificador debe ser -1, ya que nos permite un rango lo suficientemente amplio como para tener problemas en la implementación. Por otro lado se deben establecer los valores de los condensadores que son necesarios para el montaje. En este caso están condicionados únicamente por la frecuencia de conmutación, luego basta con acudir al datasheet del convertidor [13] y buscar el valor recomendado.

Para esta frecuencia son necesarios condensadores de 330µF tanto para la entrada como para la salida. Tal y como puede verse en la figura 17, incorporar estos condensadores modifica la capacidad de entrada y salida así como la ESR equivalente, puesto que cambia el condensador. Un simple cálculo nos permite obtener los nuevos valores de C y R:

$$C_{IN} = 330\mu F + 22\mu F = 352\mu F \quad \textbf{(8)}$$

$$C_{OUT} = 330\mu F + 47\mu F = 377\mu F \quad \textbf{(9)}$$

$$ESR = \frac{1{,}5m\Omega \cdot 1{,}5m\Omega}{1{,}5m\Omega + 1{,}5m\Omega} = 0{,}75m\Omega \quad \textbf{(10)}$$

Finalmente, los valores de diseño del convertidor se detallan en la tabla 6:

Tensión entrada [V]	6	**C_{i1} [µF]**	330
Tensión salida [V]	2	**C_{i2} [µF]**	22
Frecuencia [kHz]	400	**C_{O1} [µF]**	47
Corriente salida [A]	5	**C_{O2} [µF]**	330
Bobina [µH]	0,9	**ESR [µF]**	0,75
DCR [mΩ]	2,2	**Potencia salida [W]**	10

TABLA 6. VALORES DISEÑO CONVERTIDOR

4.2.1. Obtención de los parámetros del lazo de control digital empleando la metodología PowerDigit

Una de las ventajas de emplear la metodología PowerDigit es que para todos los diseños existen las mismas pautas a la hora de realizar los cálculos del lazo de control. Por ello, es objetivo del trabajo presentar de forma clara y detallada todos los aspectos relativos a esta fase del diseño. Así como las ecuaciones necesarias para su realización.

Generalmente, cuando se realiza un diseño, suelen existir limitaciones que condicionan las decisiones tomadas por el diseñador. Estas limitaciones deben ser el punto de partida de todos los cálculos, puesto que de lo contrario se podría llegar a un diseño que no cumpliera con dichas limitaciones.

En este caso, el PWM presenta una frecuencia de funcionamiento máxima de 200MHz. Este problema, detectado durante la fase de síntesis, se debe a los retrasos que introducen los componentes digitales, siendo imposible emplear relojes de frecuencia superior a la mencionada.

Por lo tanto, en este caso, el primer parámetro que deberemos calcular será cuántos pasos por ciclo de conmutación será capaz de realizar el contador del PWM teniendo en cuenta esta limitación. Lo que nos dará idea de la precisión que seremos capaces de alcanzar. Haciendo uso de la ecuación 1 obtenemos:

$$\frac{f_{CLK}}{F_{SW}} = nº\ pasos\ contador\ DPWM = \frac{200MHz}{400kHz} = \mathbf{500\ pasos}$$

Y por tanto, despejando de la ecuación 2, el número de bits máximo que podemos emplear para representar la señal PWM será:

$$n_{PWM} \geq \frac{\log_{10} 500}{\log_{10} 2} = 8{,}96 = \mathbf{9\ bits} \quad \textbf{(11)}$$

Una vez definido el número de bits es necesario calcular la resolución del PWM para este caso. Según la metodología PowerDigit y como ya comentamos, para realizar este cálculo es necesario tener en cuenta la topología del convertidor.

Ya que en nuestro caso contamos con un reductor, en primer lugar deberíamos obtener la función de transferencia del convertidor:

$$V_o = d \cdot V_i \quad \textbf{(12)}$$

Donde V_o representa la tensión de salida, d representa el ciclo de trabajo y V_i la tensión de entrada.

A partir de este resultado se debe linealizar la ecuación en torno a un punto de trabajo:

$$\Delta V_{o_{PWM}} = \left.\frac{\partial V_o}{\partial d}\right|_{V_i} = V_i \cdot \Delta d \quad \textbf{(13)}$$

Siendo Δd igual a:

$$\Delta d = \frac{1}{nº\ pasos\ contador\ PWM} = \frac{1}{500} = \mathbf{2mV} \quad \textbf{(14)}$$

Sustituyendo el resultado de 14 en 13 obtenemos la resolución en tensión del PWM:

$$\Delta V_{o_{PWM}} = V_i \cdot \Delta d = 6V \cdot (2 \cdot 10^{-3})V = \mathbf{12mV} \quad \textbf{(15)}$$

Después de calcular el número de bits y la resolución del PWM debemos calcular el número de bits y la resolución del ADC. Tal y como establece la metodología PowerDigit, es necesario que la resolución en tensión del PWM sea menor que la resolución en tensión del ADC. Además se debe tener en cuenta la ganancia del sensor para calcular $\Delta V'_{o_{ADC}}$.

En este caso, ya que la tensión de salida objetivo es medible por el ADC no haría falta colocar un bloque de sensado, sin embargo, para limitar la corriente de entrada por el pin de la FPGA de cara a las validaciones del capítulo 5 se va a colocar un divisor resistivo con ganancia muy próxima a

1 por medio de dos resistencias de 500kΩ y 1kΩ. La ganancia del sensor es la obtenida por medio de la ecuación 16.

$$G_{sensor} = \frac{R_{500}}{R_{500} + R_1} = \frac{500k\Omega}{500\text{k}\Omega + 1\text{k}\Omega} = \mathbf{0,998} \quad \textbf{(16)}$$

Donde R_{500} y R_1 representan los valores de resistencia de 500kΩ y 1kΩ respectivamente.

Una vez definida la ganancia del sensor podemos calcular el número de bits que utiliza el ADC basándonos en el resultado obtenido para el PWM. Sabemos que la resolución del ADC es:

$$\Delta V'_{O\,ADC} = \frac{\frac{Rango_{ADC}}{2^{nbitsADC}}}{G_{sensor}} > \Delta V_{o_{PWM}} = 12mV \quad \textbf{(17)}$$

Donde $Rango_{ADC}$ es la diferencia entre el valor máximo y mínimo medible, en este caso:

$$Rango_{ADC} = V_{adcMAX} - V_{adcMIN} = 2{,}9V - 0{,}4V = \mathbf{2,5V} \quad \textbf{(18)}$$

Despejando $n_{bitsADC}$ de la ecuación 20 obtenemos:

$$n_{bitsADC} = \frac{\log_{10}\left(\frac{Rango_{ADC}}{\Delta V_{o_{PWM}} \cdot G_{sensor}}\right)}{\log_{10} 2} = \frac{\log_{10}\left(\frac{2.5V}{12mV \cdot 0{,}998}\right)}{\log_{10} 2} = 7{,}7 = \mathbf{7bits} \quad \textbf{(19)}$$

Una vez obtenido el número de bits para el ADC calculamos la resolución para 7 bits.

$$\Delta V'_{O\,ADC} = \frac{\frac{Rango_{ADC}}{2^{nbitsADC}}}{G_{sensor}} = \frac{\frac{2.5V}{2^7}}{0.998} = \mathbf{19,57mV} \quad \textbf{(20)}$$

Por lo tanto, tal y como establece la metodología, se cumple que la resolución del ADC es mayor que la del PWM:

$$\Delta \boldsymbol{V'_{O\,ADC}} = \mathbf{19,57mV} > \Delta \boldsymbol{V_{o_{PWM}}} = \mathbf{12mV} \quad \textbf{(21)}$$

Como se explicará en el apartado 4.2.2, para calcular los coeficientes del lazo, la versión 9.0 de PSIM requiere de una ganancia calculada a partir del producto de las ganancias de los bloques ADC y modulador. Las siguientes ecuaciones permiten calcular dicha ganancia:

$$G_{ADC} = \frac{1}{\frac{Rango_{ADC}}{2^n - 1}} = \frac{1}{\frac{2{,}5}{2^7 - 1}} = \mathbf{50,8} \quad \textbf{(22)}$$

$$G_{PWM} = \Delta d = \frac{1}{500\ pasos} = \mathbf{2 \cdot 10^{-3}} \quad \textbf{(23)}$$

Finalmente, la ganancia del sistema será:

$$G_{total} = \prod_{i=1}^{n} G_i = G_{ADC} \cdot G_{PWM} = \mathbf{0,1016} \quad \textbf{(24)}$$

A continuación se resumen los resultados obtenidos en la siguiente tabla:

Nº bits ADC	7
Nº bits PWM	9
Resolución ADC [mV]	19,57
Resolución PWM [mV]	12
Ganancia sensor	0,998
Ganancia sistema	0,1016

TABLA 7. RESUMEN RESULTADOS DEL DISEÑO

Una vez llegados a este punto, la metodología PowerDigit requiere de una serie de pasos adicionales para diseñar el lazo. Estos pasos no están incorporados en la herramienta SmartCtrl, por lo que se explicará en detalle la obtención de todos los parámetros necesarios para diseñar los bloques funcionales.

En primer lugar, es necesario cuantificar los retrasos introducidos por la implementación digital. En este caso, los retrasos son debidos a la carga del dato de tensión en la FPGA (retraso del ADC) y del cálculo de la ecuación del regulador (retraso del regulador).

Para calcular el retraso del ADC se debe acudir al datasheet del componente [12] y calcular el número de ciclos de reloj que tarda en cargar el dato una vez ha sido muestreado.

Tal y como se vio en el apartado 4.1.2.1.2, un ciclo completo de carga representa 34 ciclos del reloj SPI_SCK.

Sin embargo, el dato muestreado tiene un ciclo de retraso, luego es necesario realizar dos lecturas del ADC para obtener el dato válido. Por lo tanto, el cálculo de este retraso sería:

$$retraso_{ADC} = t_{ADC} = 2 \cdot nº\ ciclos \cdot T_{SPI\_SCK} = 2 \cdot 34 \cdot \left(\frac{1}{f_{SPI_{SCK}}}\right) = 2 \cdot 34 \cdot \left(\frac{1}{50MHz}\right) = \mathbf{1,36\mu s} \quad \textbf{(25)}$$

Donde T_{SPI_SCK} es el período de reloj del bloque y $nº\ ciclos$ representa los ciclos necesarios para realizar una conversión.

Por último, para calcular el retraso del regulador:

$$retraso_{calculo} = t_{calculo} = nº\ ciclos \cdot \left(\frac{1}{f_{CLK}}\right) = 2 \cdot \left(\frac{1}{50MHz}\right) = \mathbf{40ns} \quad \textbf{(26)}$$

Donde el número de ciclos de la ecuación 26 representa los ciclos de reloj CLK que tarda en actualizarse el resultado tras obtener el dato por el ADC.

La tabla 8 resumen los retrasos del sistema:

Retraso ADC [µs]	1,36
Retraso regulador [ns]	40

TABLA 8. RESUMEN RESULTADOS DEL DISEÑO

Para calcular el resto de parámetros es necesario disponer de los coeficientes discretos del regulador que será implementado.

Ya que la obtención de dichos coeficientes se explicará en el apartado 4.2.2, se adelantan en este apartado los valores obtenidos para cerrar todo el proceso de cálculo.

En primer lugar, el diseñador que utilice la metodología PowerDigit deberá disponer de una hoja de cálculo prediseñada que le permita establecer los parámetros necesarios sin necesidad de conocer en profundidad la metodología.

En esta hoja de cálculo están incorporadas las condiciones que debe reunir el diseño, así como información adicional de operaciones intermedias para usuarios más avanzados.

Para clarificar el uso de este fichero, se detalla a continuación paso por paso cómo debe utilizarse.

Lo primero que se deberá rellenar es la siguiente tabla con los coeficientes en "z" obtenidos:

	Coeficientes	N bits	Reescalado		Redondeo	
			1			
b0			0		0	
b1			0		0	
b2			0		0	
b3			0		0	0
	0					
a0			0		0	
a1			0		0	
a2			0		0	
a3			0		0	0
	0					

FIGURA 25. PLANTILLA DE OBTENCIÓN DE COEFICIENTES RE-ESCALADOS

Una vez rellena se deberá calcular el parámetro N bits que depende del número de decimales que elijamos representar. Cómo obtener este parámetro se explica a través del ejemplo mostrado en la figura siguiente:

	Coeficientes	N bits	Reescalado		Redondeo	
		24	16777216			
b0	3.6898289		61905056.46		61905056	
b1	-2.5699004		-43115774.11		-43115774	
b2	-3.6048493		-60479335.35		-60479335	
b3	2.65488		44541495.21		44541495	2851442
	0.1699592					
a0	1		16777216		16777216	
a1	-0.5864966		-9839780.141		-9839780	
a2	-0.37075651		-6220262.052		-6220262	
a3	-0.04274689		-717173.7565		-717174	0
	3E-09					

FIGURA 26. EJEMPLO DE OBTENCIÓN DE LOS COEFICIENTES RE-ESCALADOS

Vemos que se ha elegido emplear 24 bits para representar 7 decimales. Para obtener este número a partir de los decimales a representar se utiliza la siguiente fórmula:

$$N^{\underline{o}}{}_{bits} > \frac{\log_{10}(10^{n_{dec}})}{\log_{10} 2} = \frac{\log_{10}(10^{7})}{\log_{10} 2} = 23{,}25 \rightarrow N^{\underline{o}}{}_{bits} = \mathbf{24\ bits} \quad (27)$$

Donde n_{dec} representa el número de decimales que se quieren conservar para la implementación digital.

En la figura puede verse como en la columna de redondeo aparecen los coeficientes sin decimales ya re-escalados. Para comprobar que todo es correcto, el valor del recuadro naranja en los coeficientes "*b"* debe ser distinto de cero. Por el contrario, para los coeficientes "*a"* debe ser cero.

Este resultado nos proporciona el valor de los parámetros KE y KD, necesarios para la implementación del lazo. Sin embargo, el parámetro KE debe cumplir otra condición. Según la metodología PowerDigit es necesario que el sumatorio de los coeficientes re-escalados de "*b"* sea mayor que el valor 2^{nbits} obtenido anteriormente [1]. Para realizar esta modificación se emplea la figura 27:

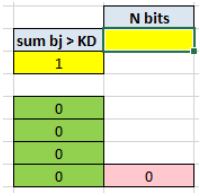


FIGURA 27. CORRECCIÓN DEL PARÁMETRO KE

El diseñador deberá rellenar el recuadro amarillo vacío hasta que el recuadro rosa se ponga verde oscuro, una vez en ese punto, estamos en condiciones de obtener un diseño válido. Los nuevos coeficientes de *"b"* que deberán usarse se mostrarán en los recuadros verde claro. Siguiendo con el ejemplo anterior:

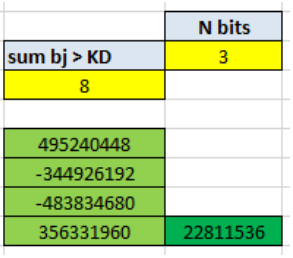


FIGURA 28. EJEMPLO DE CORRECCIÓN DEL PARÁMETRO KE

Por último, se debe cumplir la siguiente condición [1]:

$$n^{\underline{o}}bits\ KD + n^{\underline{o}}bits\ decimales\ \geq n^{\underline{o}}bits\ KE \quad (28)$$

Siendo nº bits decimales la representación decimal del ciclo de trabajo calculado. Para el caso de este diseño, el número de bits decimales elegido es:

$$n^{\underline{o}}bits\ decimales = \mathbf{5}\ \geq (n^{\underline{o}}bits\ KE - n^{\underline{o}}bits\ KD) = 27 - 24 = 3 \quad (29)$$

Una vez obtenidos estos parámetros se da por finalizado el diseño del lazo completo. Para resumir todos los resultados obtenidos se presenta la siguiente tabla:

Nº bits ADC	7
Nº bits PWM	9
Resolución ADC [mV]	19,57
Resolución PWM [mV]	12
Ganancia sensor	0,998
Ganancia sistema	0,1016
Retraso ADC [µs]	1,36
Retraso regulador [ns]	60
Nº bits KE	27
Nº bits KD	24
Nº bits decimales	5

TABLA 9. RESUMEN DE LOS PARÁMETROS DEL LAZO DE CONTROL

La tabla incluye todos los parámetros que se encuentran en los bloques funcionales de PSIM y MATLAB. Por lo que, en este punto, el diseñador podría ser capaz de realizar simulaciones para validar el diseño en bloques funcionales.

4.2.2. Obtención de los coeficientes del regulador empleando la herramienta SmartCtrl

Una vez descritos los pasos a realizar para obtener todos los parámetros de los bloques funcionales, en este apartado, se va a detallar cómo obtener los coeficientes del regulador que será implementado. Para ello, se empleará la herramienta SmartCtrl, que está integrada en la versión 9.0 de PSIM. En el caso de utilizar otras versiones posteriores, la única diferencia radica en cómo configurar la ganancia del lazo, que debe configurarse utilizando otros parámetros.

Una vez abierta la herramienta, se selecciona el tipo de topología del convertidor y el tipo de lazo de control. En nuestro caso, convertidor reductor controlado por un lazo simple en modo tensión.

A continuación, el programa nos pedirá introducir las especificaciones del convertidor. Además, nos proporciona información sobre cuál es el ciclo de trabajo en régimen permanente y la corriente media y de pico por la bobina.

En la figura 29 puede verse una captura de la herramienta en esta fase.

Una vez se termine de especificar el convertidor, el siguiente paso será elegir el sensor. En nuestro caso es un divisor resistivo cuyos valores fueron calculados en el apartado 4.2.1.

La pantalla que muestra Smart Control se muestra en la en la figura 30.

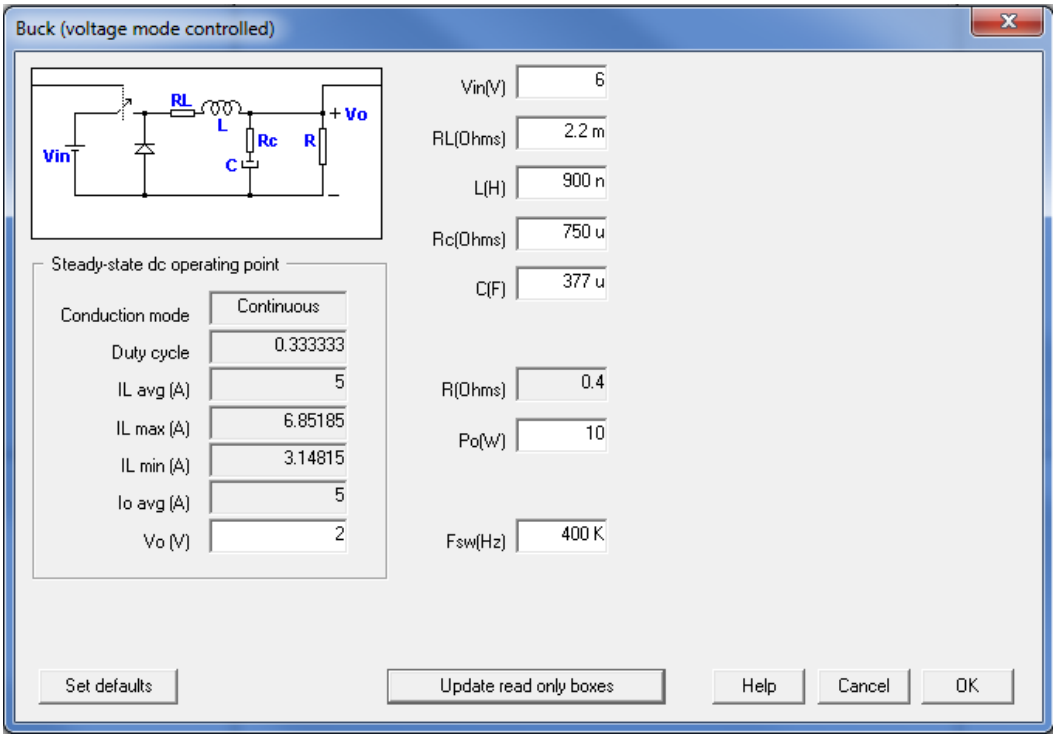


FIGURA 29. ESPECIFICACIONES DEL CONVERTIDOR PARA SMARTCTRL

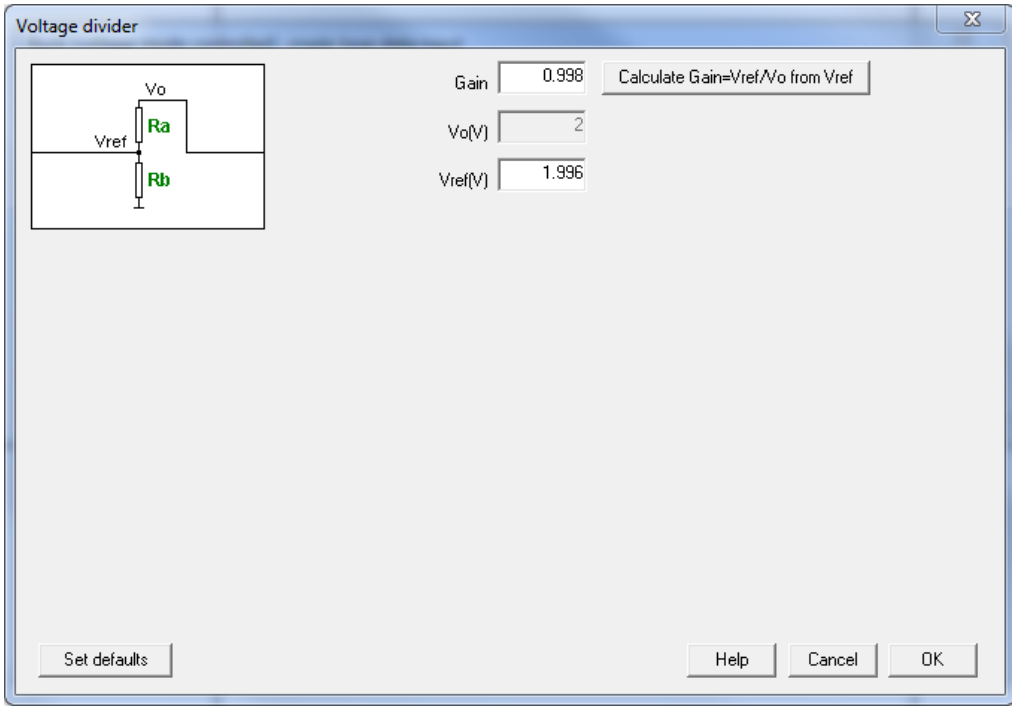


FIGURA 30. ESPECIFICACIONES DEL SENSOR EN SMARTCTRL

El último bloque que debemos configurar es el del regulador. En nuestro caso emplearemos un regulador tipo 3. Se ha decidido este tipo de regulador ya que se utiliza para sistemas de segundo orden y proporciona la mejor respuesta dinámica de todos los disponibles en la herramienta. La función de transferencia en el dominio "z" de dicho regulador es la siguiente (ecuación 30):

$$R(z) = \frac{(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3})}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} = \frac{b_0 z^3 + b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3} \quad (30)$$

El diseñador será el encargado de elegir el tipo de regulador más conveniente para su sistema. La metodología PowerDigit no condiciona el uso de un tipo de regulador específico ya que el diseño está preparado para implementar cualquier tipo de regulador.

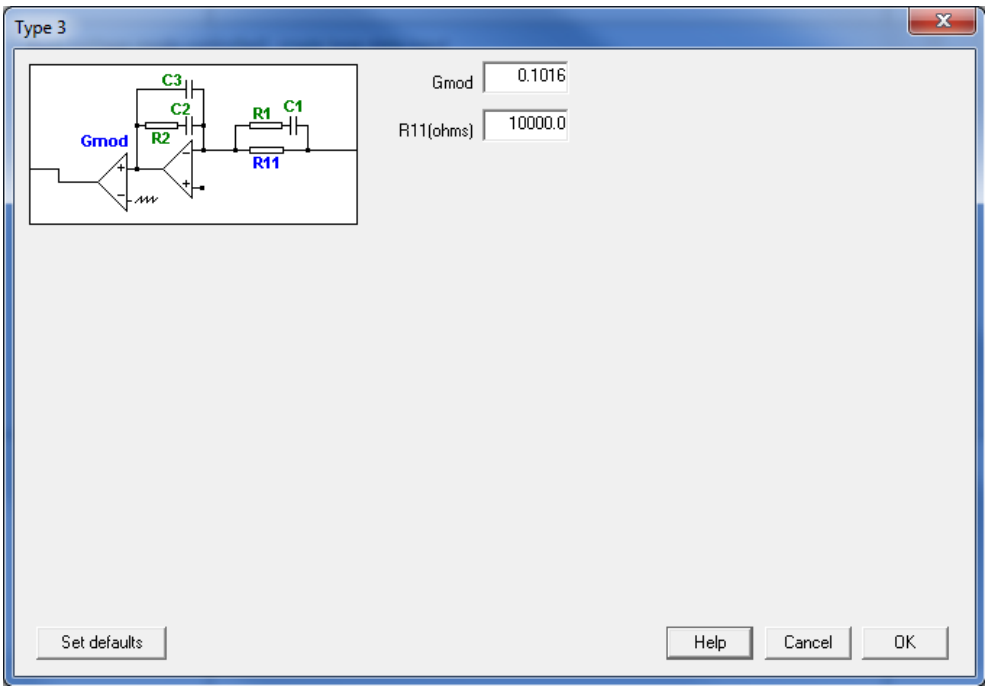


FIGURA 31. ESPECIFICACIONES REGULADOR EN SMARTCTRL

Una vez configurados todos los bloques, SmartCtrl nos presenta un mapa de soluciones como el que muestra la figura 32:

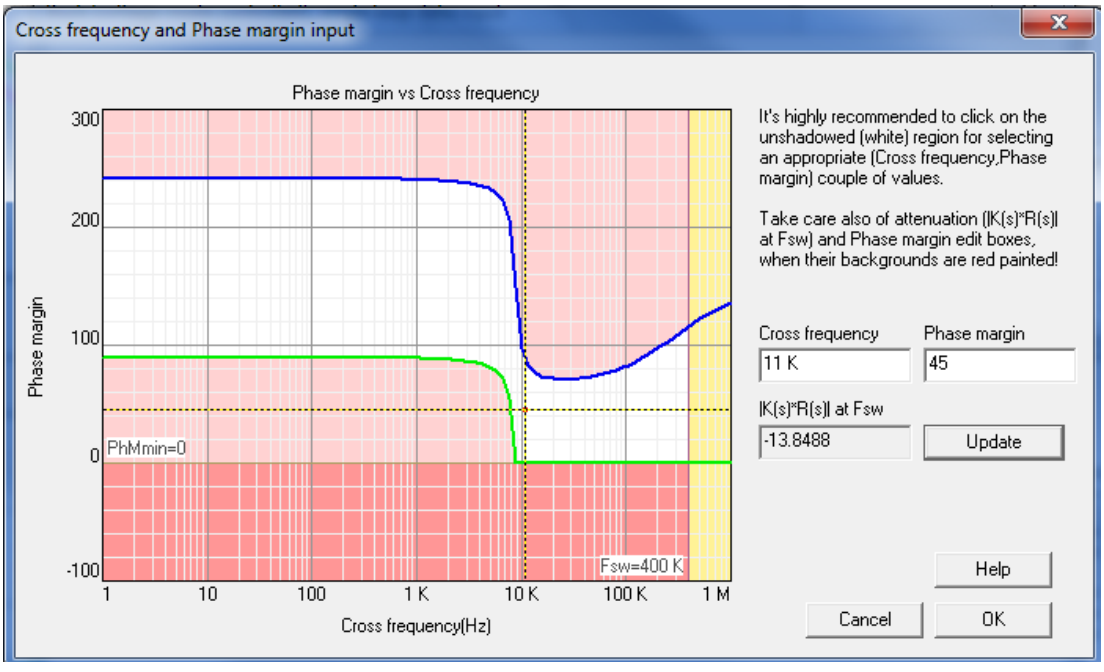


FIGURA 32. MAPA DE SOLUCIONES DE SMARTCTRL

Cualquier diseño dentro de la zona blanca es válido, sin embargo, se deben tener en cuenta las especificaciones dinámicas del sistema para elegir un punto adecuado. En nuestro caso, se ha elegido una frecuencia de cruce (f_c) de 11kHz y un margen de fase (MF) de 45º.

El motivo de elegir esa frecuencia de cruce se debe al criterio que establece [14], por el cual, para tener una atenuación suficiente del rizado de conmutación es necesario que la frecuencia de cruce sea del orden de una década menor que la frecuencia de conmutación. Además, debe ser lo suficiente alta para tener una buena regulación dinámica.

Por otro lado, el MF deberá ser suficiente para proporcionar una respuesta amortiguada y dar seguridad ante posibles desfases adicionales, y suficientemente bajo para dar buena regulación dinámica [14].

Los valores típicos del MF oscilan entre [30-90] º, siendo 45º una elección adecuada pasa este caso. El lector que desee profundizar más acerca de esta elección puede acudir a la referencia [15] de la bibliografía.

Una vez se ha terminado de configurar todo lo anterior, SmartCtrl nos presentará diversas gráficas con los resultados obtenidos tras el diseño.

Entre ellos se encuentran diagramas de Bode de las funciones de transferencia más importantes, un diagrama de estabilidad de Nyquist y una gráfica de la respuesta temporal.

La figura 33 representa la tensión de salida del convertidor ante un escalón de tensión en la referencia de tensión:

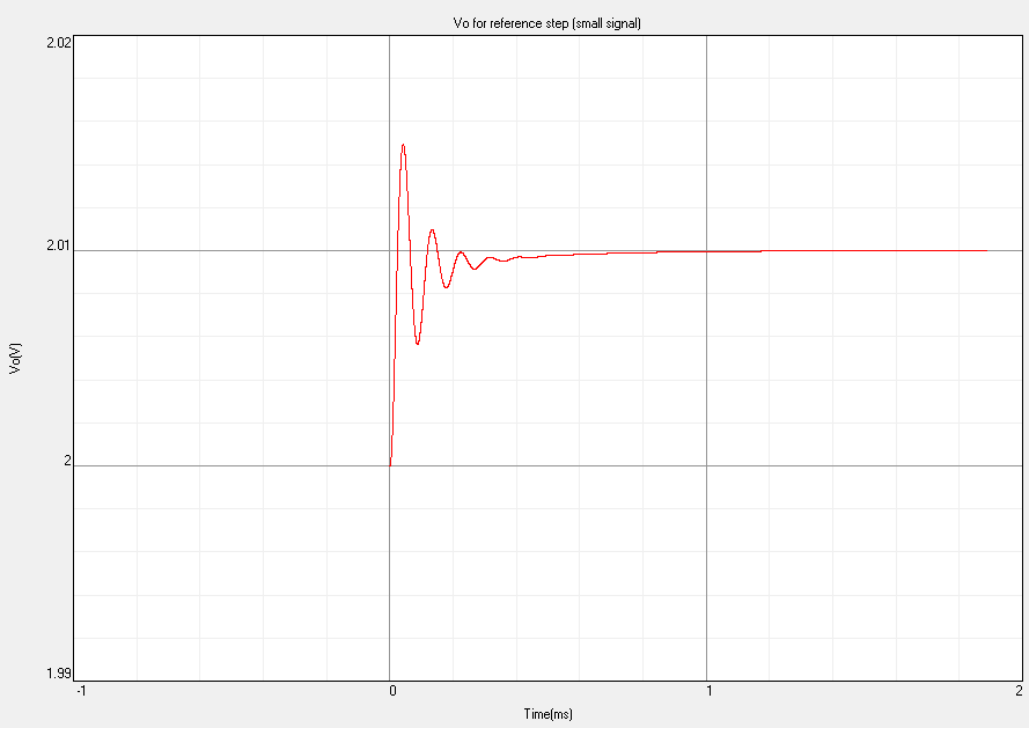


FIGURA 33. RESPUESTA ANTE ESCALÓN EN LA REFERENCIA DEL SISTEMA EN SMARTCTRL

Como vemos, el sistema permanece estable tras la perturbación. Por lo que los parámetros de diseño escogidos son válidos.

Una vez finalizado el diseño del regulador, SmartCtrl permite obtener numerosos detalles de la implementación: Componentes analógicos necesarios para implementar el regulador, frecuencias características de cada elemento, exportar los resultados a la hoja de esquemáticos de PSIM, etc. Sin embargo, lo único que nos interesa para aplicar la metodología PowerDigit son los coeficientes discretos del regulador, los coeficientes *"z"*.

La versión empleada de SmartCtrl sólo proporciona los coeficientes en continuo, coeficientes en *"s"*, por lo que es necesario discretizarlos usando la herramienta s2z que está disponible en PSIM; o bien empleando otro tipo de herramientas como MATLAB.

La figura 34 muestra la discretización de los coeficientes empleados para el diseño realizado en el apartado anterior:

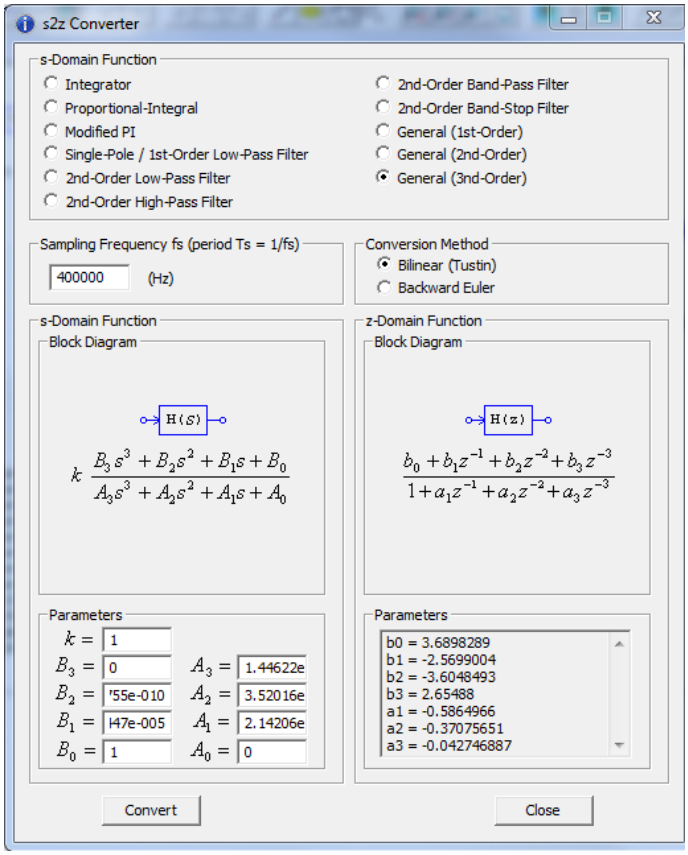


FIGURA 34. DISCRETIZACIÓN DE LOS COEFICIENTES DEL REGULADOR USANDO S2Z DE PSIM

Finalmente, la expresión del regulador discretizado es:

$$R(z) = \frac{(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3})}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} = \frac{b_0 z^3 + b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3}$$

$$= \frac{\mathbf{3,6898289 z^3 - 2,5699004 z^2 + -3,6048493 z + 2,65488}}{\mathbf{z^3 - 0,5864966 z^2 - 0,37075651 z - 0,042746887}} \quad \textbf{(31)}$$

Una vez llegados a este punto, sólo es necesario introducir los coeficientes obtenidos en la hoja de cálculo del apartado 4.2.1, y continuar desde ahí con los pasos que se describen.

4.3. Aspectos prácticos

Una vez explicados los elementos que van a constituir el sistema y definidos todos los parámetros necesarios, se van a detallar los bloques VHDL que serán usados finalmente para la implementación digital y que están disponibles en la metodología PowerDigit.

Al igual que sucedía con el modelo en bloques funcionales, la metodología PowerDigit sólo contaba con un bloque ADC, el cual no requería de ningún bloque adicional más allá del controlador. Si bien es cierto que cada ADC requerirá de su controlador específico, hay algunos bloques adicionales que pueden reutilizarse de unos a otros en caso de ser necesario.

Por ejemplo, en el caso del apartado 3.1 se comentó la posibilidad de que los ADC dispongan de una ganancia y un offset para que funcionen correctamente, esto supone que es posible añadir un bloque amplificador que introduzca una ganancia programable al ADC en función del diseño.

El diagrama de bloques que realiza este bloque será el siguiente (figura 35):

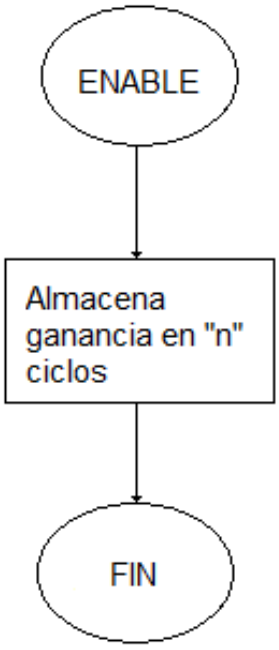


FIGURA 35. DIAGRAMA DE BLOQUES AMPLIFICADOR GENÉRICO

A través de unos parámetros configurables se podrá definir cuantos ciclos de carga son necesarios para actualizar la ganancia. Requiere de una señal de habilitación para comenzar a funcionar y devuelve un pulso de finalización cuando termina de almacenar la ganancia en el registro correspondiente.

Para el caso en el que nos encontramos será necesario disponer de un bloque amplificador como el comentado, puesto que tal y como se vio en el apartado 4.2, el dato es multiplicado por una

ganancia configurable. Para solventar este problema basta con hacer uso del bloque genérico disponible y configurar sus genéricos para adaptarlo a las necesidades de este ADC.

Por lo tanto, haciendo uso del diseño de genéricos planteado en el capítulo 3, modificamos los parámetros relativos al amplificador como vemos en la figura 36:

```
generic(    g_steps_amp: integer := 8;
            g_nbits_c_amp: integer := 3;
            g_nbits_r_amp: integer := 8);
```

FIGURA 36. GENÉRICOS BLOQUE AMPLIFICADOR

Donde *"g_steps_amp"* define el número de ciclos de carga, *"g_nbits_c_amp"* define el número de bits necesarios para el contador binario y "*g_nbits_r_amp"* define el número de bits del registro de almacenamiento de la ganancia.

Además, en algunos casos es necesario gestionar el bloque controlador y el amplificador a distintas frecuencias de reloj, por lo que también se dispone de un bloque temporizador que permite configurar la frecuencia de reloj en función del elemento que actúe.

El diagrama de bloques del temporizador puede verse en la figura 37:

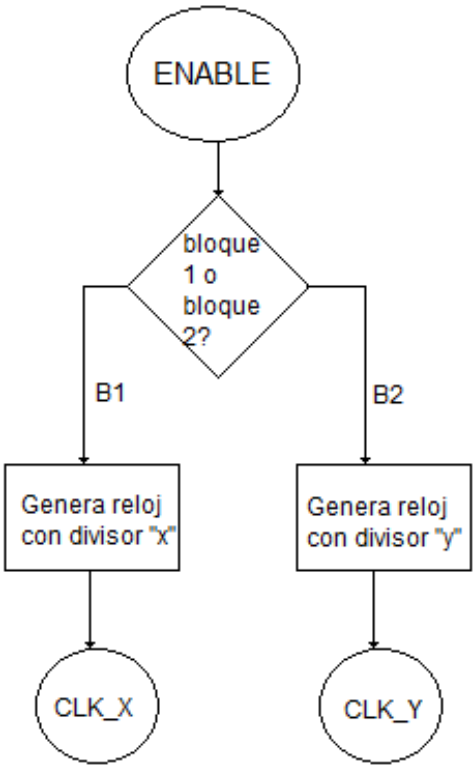


FIGURA 37. DIAGRAMA DE BLOQUES TEMPORIZADOR GENÉRICO

El temporizador recibe una habilitación que en función de su valor le permite elegir el reloj que debe generar. Para cada caso el reloj generado dependerá de la configuración previa que se haga del bloque. Finalmente el temporizador devolverá a su salida un reloj u otro.

En este caso, tal y como se lee en la hoja de características de la FPGA [12], la frecuencia de reloj del amplificador y del ADC pueden ser distintas.

Debido a que el amplificador trabaja a un máximo de 10MHz, y que los retrasos calculados no permiten realizar una conversión digital por ciclo de conmutación a esa frecuencia, es necesario incluir el bloque temporizador que seleccione la frecuencia adecuada en función de qué bloque esté en funcionamiento.

Este bloque configurable permite establecer la frecuencia de dos bloques cualesquiera modificando el siguiente fragmento de código (figura 38):

```
generic(  g_nbits_cnt: integer := 5;
          g_steps_dv_adc: integer := 0;
          g_steps_dv_amp: integer := 4);
```

FIGURA 38. GENÉRICOS BLOQUE TEMPORIZADOR

Donde *"g_nbits_cnt"* define el número de bits del contador digital, *"g_steps_dv_adc"* define el valor por el que divides la frecuencia del reloj que va al ADC respecto del reloj de sistema, y *"g_steps_dv_amp"* define lo mismo que el anterior pero para el bloque amplificador.

Siguiendo esta configuración los relojes de cada bloque serán:

Para el caso del amplificador, la velocidad máxima de transferencia puede ser, como ya vimos, de 10MHz. En el diseño genérico funciona a la velocidad máxima ya que:

$$\frac{100MHz}{10\ ciclos} = 10MHz \qquad (32)$$

En el caso del ADC la velocidad máxima de transferencia son 51MHz. En este caso se emplearán 50MHz:

$$\frac{100MHz}{2\ ciclos} = 50MHz \qquad (33)$$

Los 10 y 2 ciclos son debidos a los genéricos *"g_steps_dv_amp"* y *"g_steps_dv_adc"*, respectivamente.

Finalmente, tras añadir los bloques comentados, una vista esquemática de la arquitectura a implementar es la mostrada en la figura 39.

El bloque DCM es el encargado de generar los relojes que controlarán el flujo de los controladores de ADC y de control. Esto se debe a la escasa resolución que podría alcanzarse con el reloj interno de la placa, debido a lo cual, como ya se comentó en el apartado 3.3.1, es necesario aumentar la frecuencia del reloj usando el DCM.

Por otra parte vemos como el controlador de ADC está compuesto por el bloque ADC específico de cada modelo y los bloques TEMP y AMP que son configurables y reutilizables para otros diseños.

Por último, el bloque de control está compuesto por el regulador y el modulador, que serán los encargados de calcular el ciclo de trabajo y generar la señal de disparo PWM respectivamente.

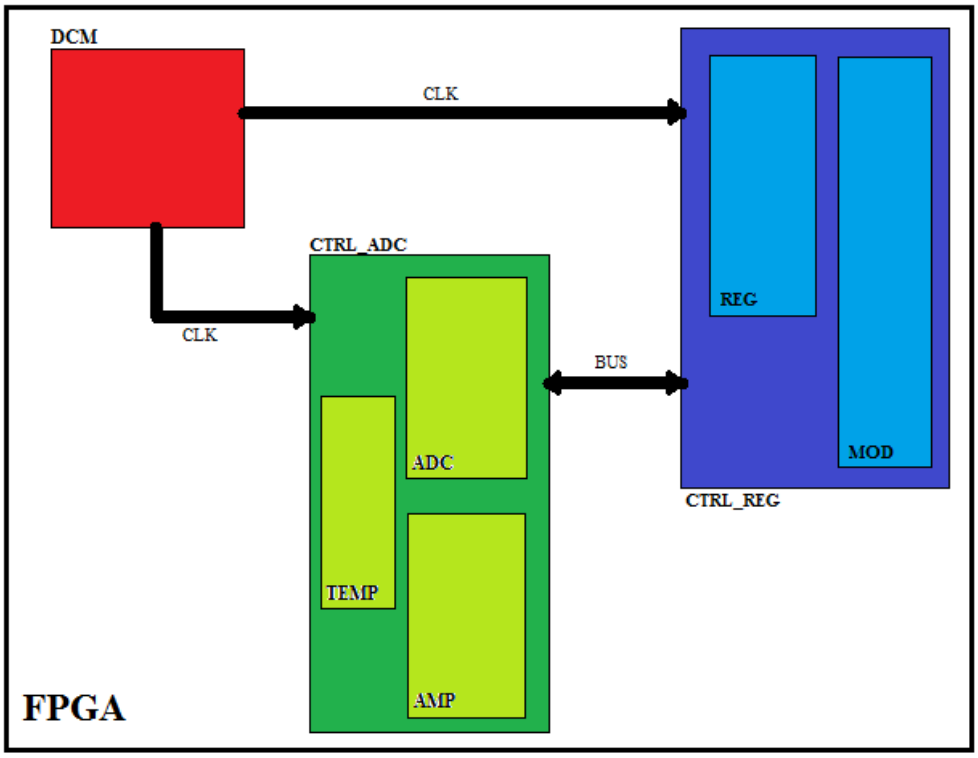


FIGURA 39. DIAGRAMA DE BLOQUES ARQUITECTURA DEL DISEÑO

5. VALIDACIÓN

5.1. Validación del diseño realizado en simulación

En este apartado se muestran los resultados obtenidos tras las simulaciones realizadas en PSIM y MATLAB. Además, se hace una comparativa de los resultados de tal forma que el lector sea capaz de apreciar fácilmente las posibles diferencias existentes entre los modelos y la implementación. Por último, se analizan en detalle los resultados y las posibles vías de mejora del diseño.

5.1.1. Comparativa de los resultados obtenidos entre PSIM y MATLAB

Como ya se expuso en el capítulo 1. Uno de los objetivos del presente trabajo es demostrar que la metodología PowerDigit es independiente de la herramienta de diseño que se utilice. En este caso, se ha desarrollado un nuevo modelo en bloques funcionales empleando MATLAB, tal y como se detalló en el capítulo 3. A continuación, se presentan los resultados de simulación del diseño en PSIM frente al diseño en MATLAB.

5.1.1.1. Respuesta en régimen transitorio

La simulación de la figura 40 muestra el arranque de ambos sistemas hasta alcanzar el régimen permanente.

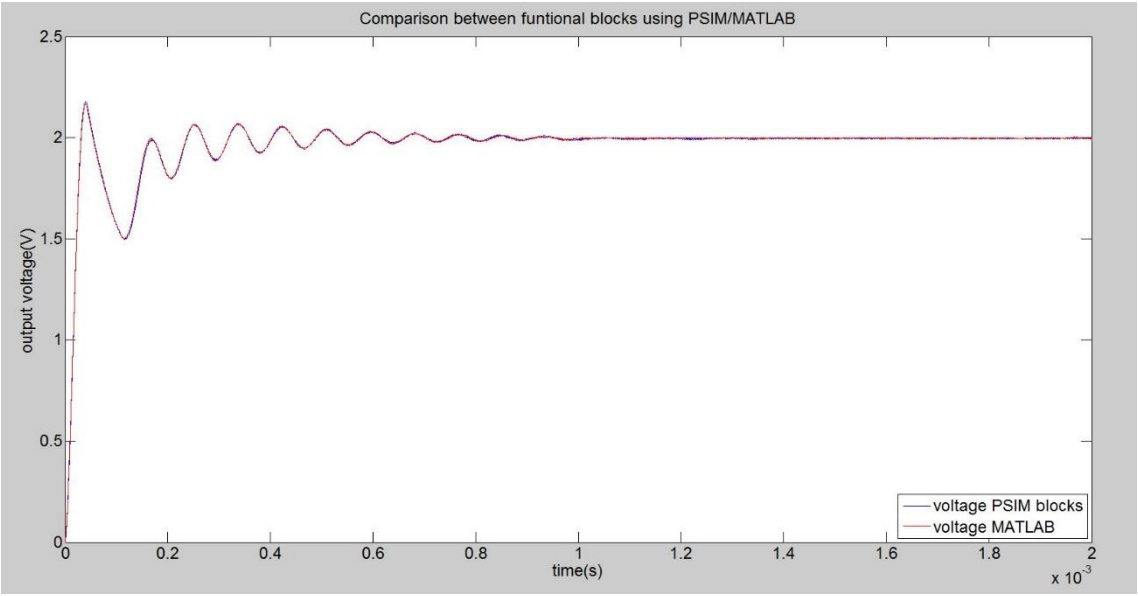


FIGURA 40. COMPARACIÓN ENTRE BLOQUES FUNCIONALES PSIM VS MATLAB

Vemos que los dos modelos tienen el mismo comportamiento dinámico, ya que la sobreoscilación y el tiempo de establecimiento de la señal es el mismo. Una vista en detalle se muestra en la figura 41.

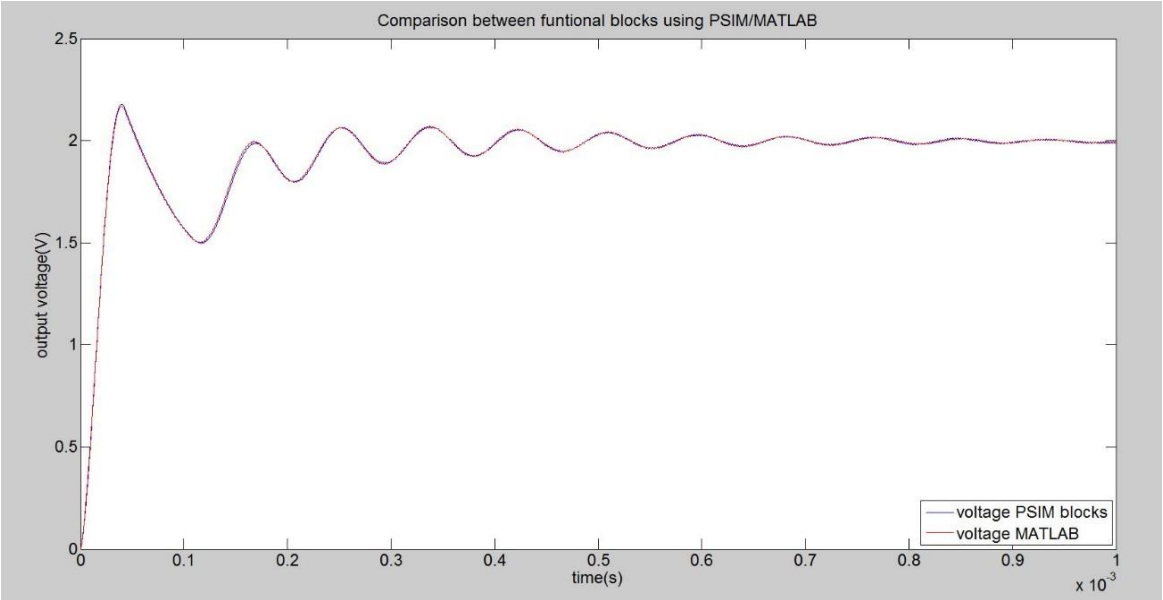


FIGURA 41. ARRANQUE SISTEMA PSIM VS MATLAB

Tal y como se aprecia en la figura 41, la vista en detalle del arranque confirma que ambos modelos son equivalentes en el régimen transitorio; y por tanto, que tienen similares prestaciones dinámicas.

5.1.1.2. Respuesta en régimen permanente

La figura 42 muestra una vista detallada del valor en régimen permanente de los dos modelos.

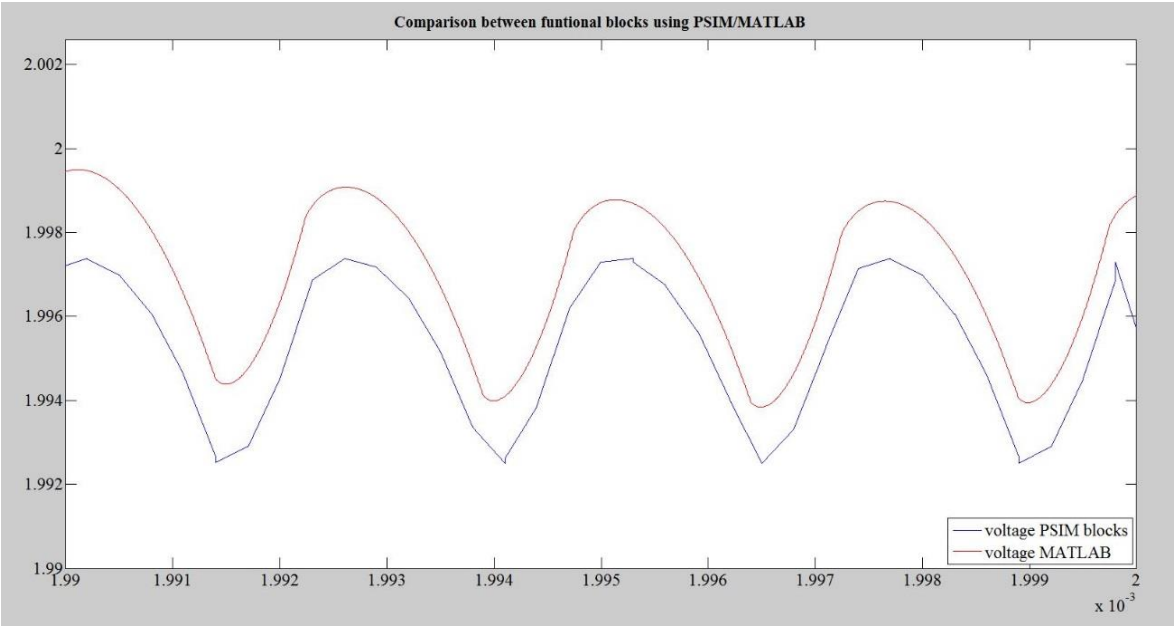


FIGURA 42. VISTA DETALLADA RÉGIMEN PERMANENTE PSIM VS MATLAB

Se aprecia como el desfase entre ambas señales es nulo y que existe una variación en el valor final de 1.5mV.

Estos resultados confirman que ambos modelos tienen el mismo retraso y que el error es mucho más pequeño que la resolución del ADC, que es de 20mV. Por lo que no debe considerarse significativo.

5.1.1.3. Respuesta ante variación en la carga

Para validar completamente la equivalencia entre los dos modelos es necesario introducir una perturbación en alguno de los parámetros de los que depende el sistema. La tensión de entrada y la carga son sensibles a variaciones durante el funcionamiento del convertidor, por lo que son los posibles parámetros a perturbar. En este caso, se ha decidido introducir variaciones en la carga. Las simulaciones realizadas son las siguientes:

- Variación de 0.4Ω a 0.2 Ω. Potencia de salida de 10W a 20W

- Variación de 0.4Ω a 0.3 Ω. Potencia de salida de 10W a 15W.

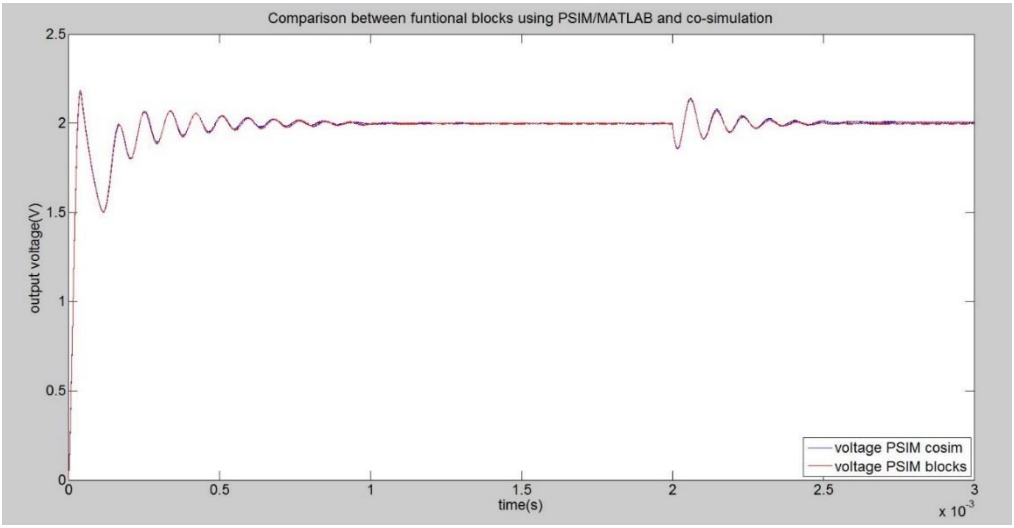


FIGURA 43. SIMULACIÓN CON VARIACIÓN EN LA CARGA A 0.2Ω

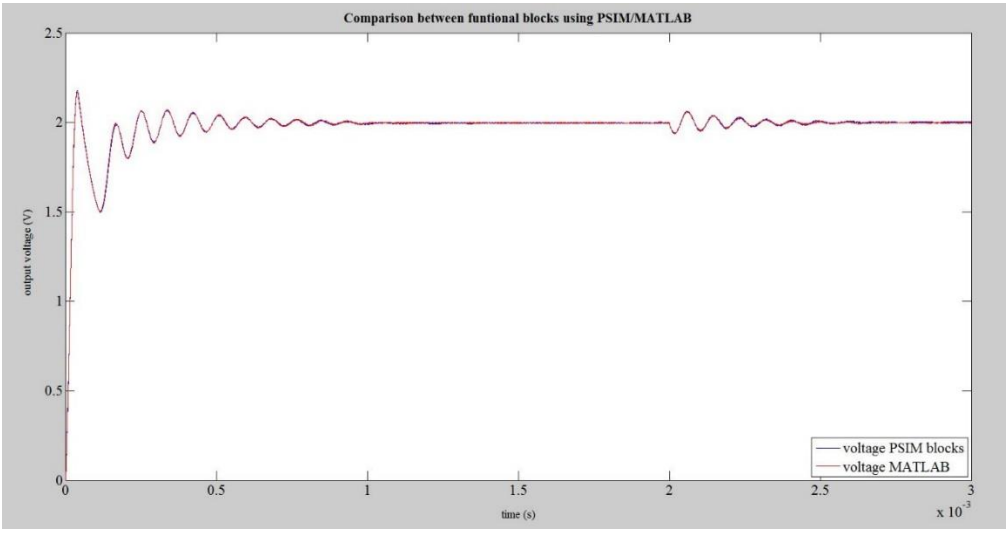


FIGURA 44. SIMULACIÓN CON VARIACIÓN EN LA CARGA A 0.3Ω

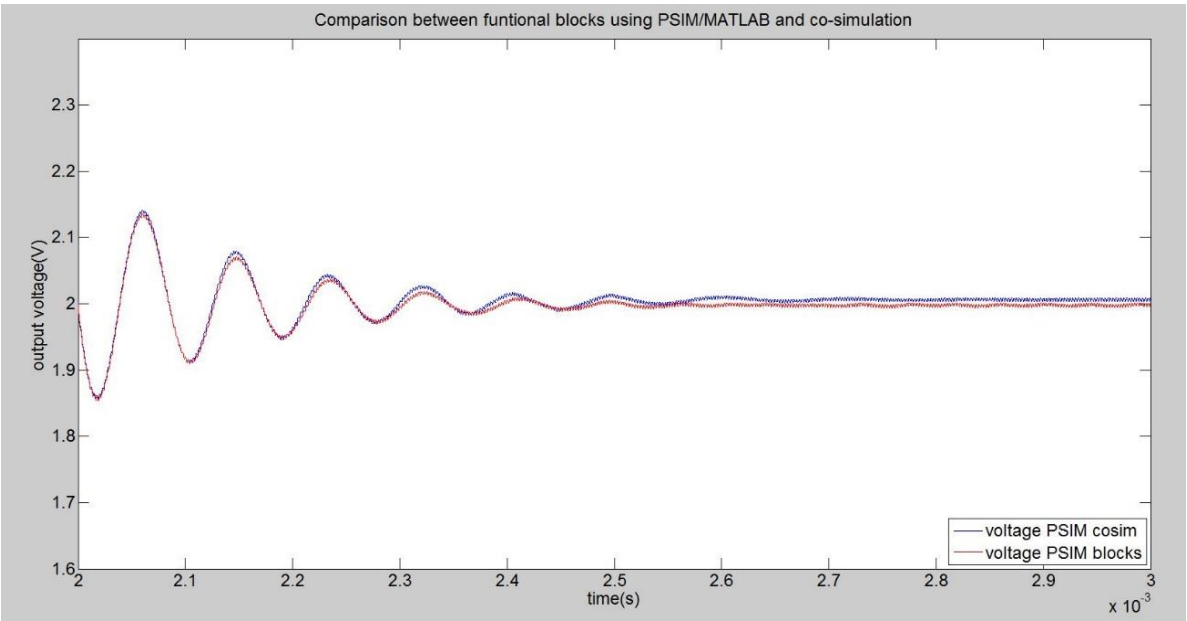


FIGURA 45. VISTA DETALLADA RESPUESTA TRANSITORIA TRAS VARIACIÓN EN LA CARGA

Los resultados obtenidos en las figuras 43, 44 y 45 muestran como el comportamiento de ambos modelos es similar tras introducir una variación en el sistema.

Esto confirma la conclusión obtenida tras el arranque; ambos modelos tienen características dinámicas similares.

Por otra parte, el error en régimen permanente es mucho menor que la resolución del ADC y del PWM, estos resultados muestran que el diseño es válido, y que por tanto, sería posible realizar una validación experimental.

5.2. Validación del diseño hardware a implementar en la FPGA

Para garantizar que el código que es empleado en co-simulación cumple con los requisitos de diseño y que no existen errores de codificación se ha realizado un banco de pruebas que permite enviar datos al sistema por medio del SDATA.

Otro motivo por el cual se realizan estas comprobaciones es la numerosa cantidad de modificaciones que ha sufrido el diseño respecto al empleado en la referencia bibliográfica [1].

Estas modificaciones no se deben únicamente a los bloques añadidos, y que fueron descritos en el capítulo 3, sino por la utilización de distintos relojes en el sistema. Esto conlleva modificar la generación de algunas señales para que puedan ser leídas por los bloques que utilizan un reloj más lento; como el caso del regulador.

En primer lugar, la figura 46 muestra los ciclos de carga del registro de la ganancia del amplificador.

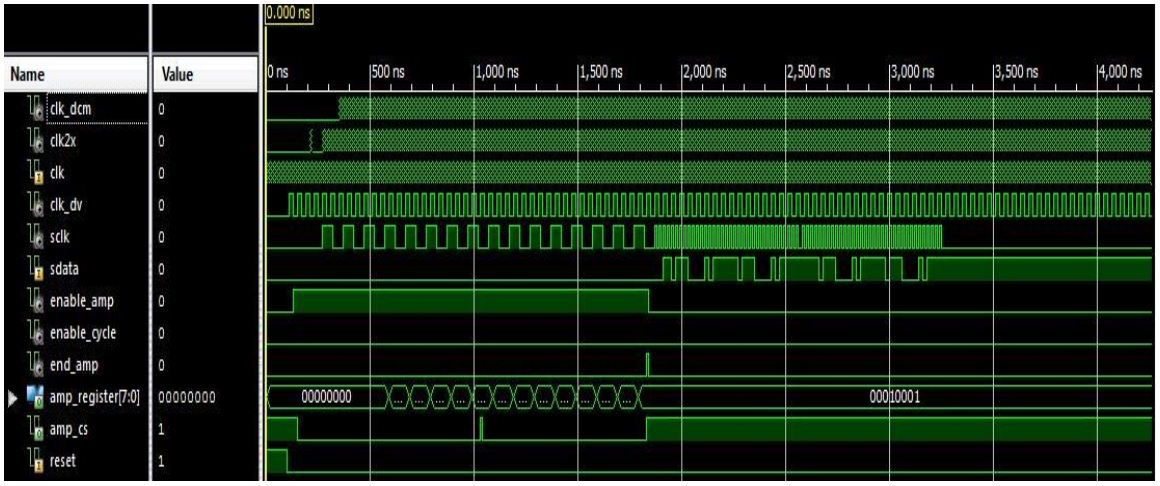


FIGURA 46. SIMULACIÓN CICLOS DE CARGA REGISTRO DEL AMPLIFICADOR

Tras realizar el ciclo completo de carga, la ganancia del *amp_register* muestra el valor correspondiente a la ganancia -1 (*"00010001"*); que es la necesaria para el caso de este diseño.

En la figura 47 se muestra la carga del canal 0 del ADC (*channel_0*) y como se realiza el cambio de coordenadas (*sample_data*).

Además, pueden verse las señales que comunican el bloque controlador ADC con el bloque regulador (*end_adc* y *en_control*).

Estas dos señales, como ya se comentó, requieren ser generadas durante varios ciclos del reloj *clk2x* (100MHz) para que puedan ser leídas por el regulador que utiliza un reloj más lento (*clk_dv @ 25MHz*).

El *"testbench"* realizado hace 4 envíos al ADC. Los resultados esperados y los obtenidos se muestran en la tabla 10.

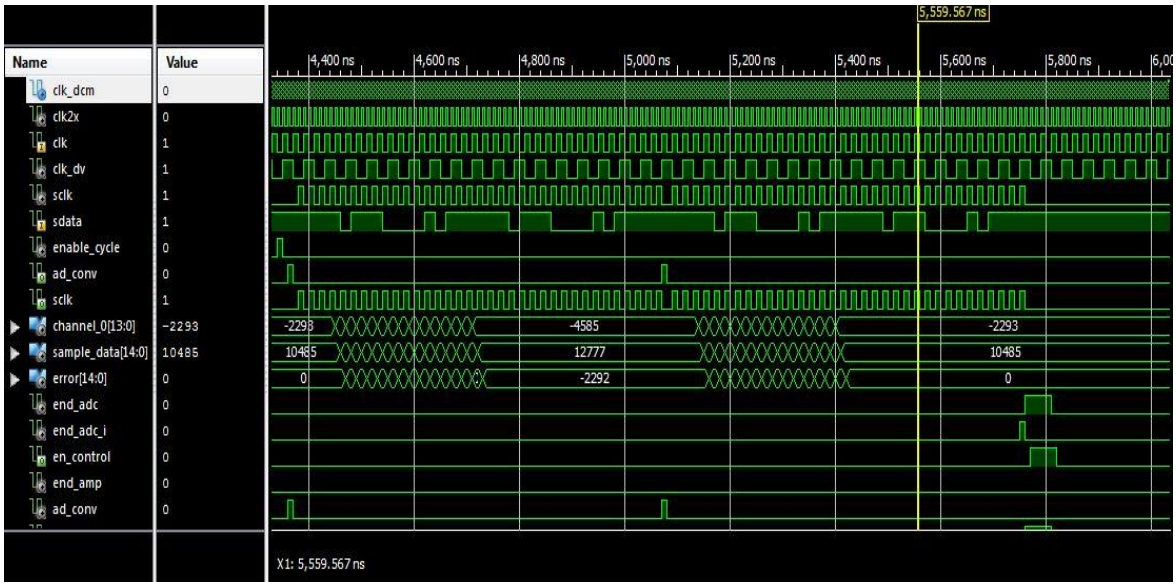


FIGURA 47. SIMULACIÓN CICLOS DE CARGA EN EL REGISTRO DEL CANAL 0

Valor analógico [V]	Dato esperado CH0	Dato obtenido CH0
2,0	-2293	-2293
2,1	-2949	-2949
1,9	-1638	-1638
1,5	983	983

TABLA 10. RESULTADOS OBTENIDOS CARGA DEL CANAL 0 DEL ADC

Por lo tanto, a la vista de los resultados obtenidos en la tabla 10, se confirma que las incorporaciones al diseño funcionan correctamente y el diseño puede ser utilizado en co-simulación con garantías de éxito.

5.3. Validación del diseño realizado en co-simulación

5.3.1. Comparativa de los resultados obtenidos entre simulación y co-simulación

Para validar los resultados de los modelos frente a la implementación hardware, al igual que en el apartado anterior, se han realizado simulaciones del arranque, régimen permanente y respuesta a perturbación.

La co-simulación se ha realizado desde PSIM, utilizando el bloque *ModCoupler* que permite conectar la planta analógica con el código VHDL. Se ha desarrollado un manual de configuración y utilización del bloque en los anexos para aquellos usuarios que se encuentren con problemas derivados de la utilización de dicho bloque.

5.3.1.1. Respuesta en régimen transitorio

La figura 48 muestra la respuesta en régimen transitorio de los tres sistemas.

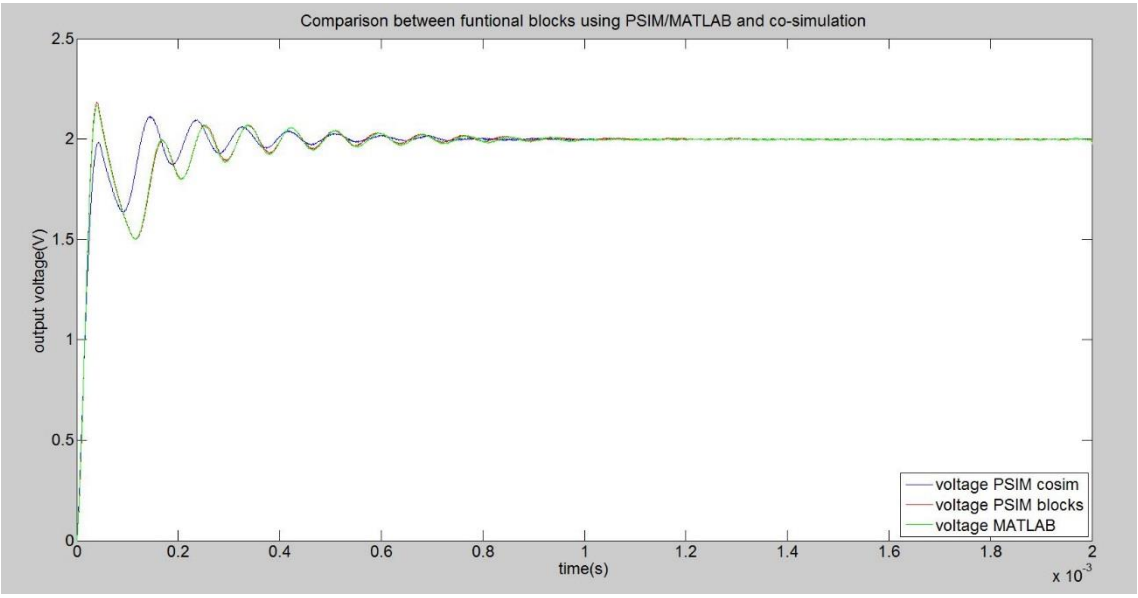


FIGURA 48. COMPARATIVA RESULTADOS SIMULACIÓN VS CO-SIMULACIÓN

Vemos que existen diferencias en el arranque entre los modelos de bloques funcionales y la implementación hardware.

Una vista detallada de estas diferencias puede verse en la figura 49.

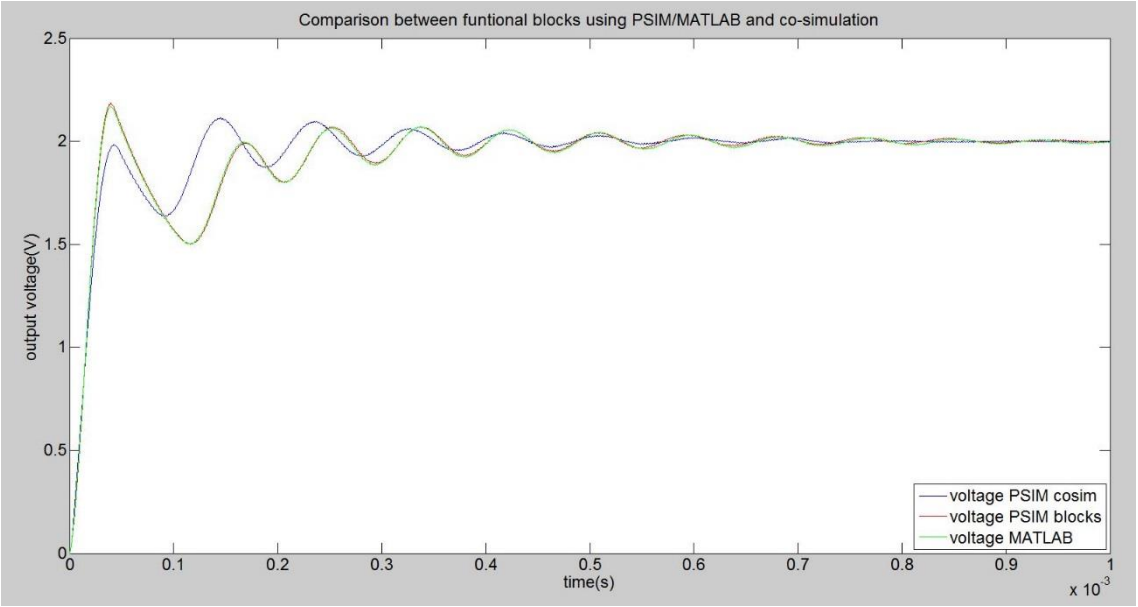


FIGURA 49. COMPARATIVA RESULTADOS SIMULACIÓN VS CO-SIMULACIÓN EN EL ARRANQUE

Desde el punto de vista dinámico, el sistema hardware tiene una respuesta más rápida y una sobreoscilación más pequeña, lo que implica que su respuesta dinámica se ha modificado respecto al diseño en bloques funcionales.

5.3.1.2. Respuesta en régimen permanente

La figura 50 muestra los resultados obtenidos en régimen permanente.

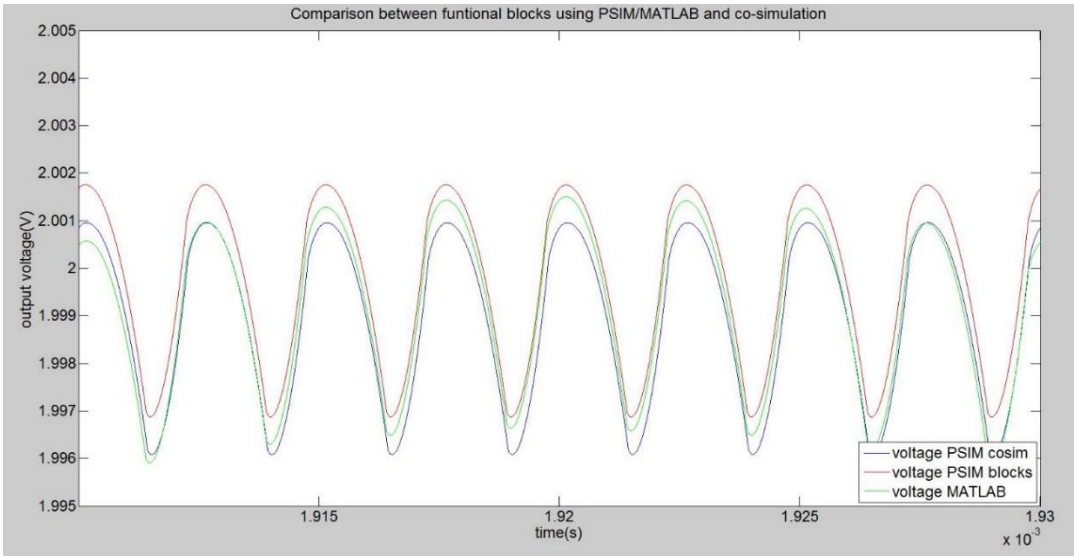


FIGURA 50. VISTA EN DETALLE DEL RÉGIMEN PERMANENTE EN LOS TRES SISTEMAS

Pese a tener una respuesta dinámica algo distinta, los modelos funcionales alcanzan el mismo valor final que el sistema de co-simulación, con una variación de 1,5mV y sin desfase.

Esto confirma que el sistema hardware mantiene las mismas características estáticas que los modelos funcionales.

5.3.1.3. Respuesta ante variación en la carga

Para confirmar la variación en la respuesta dinámica se introduce una variación en la carga como representa la figura 51.

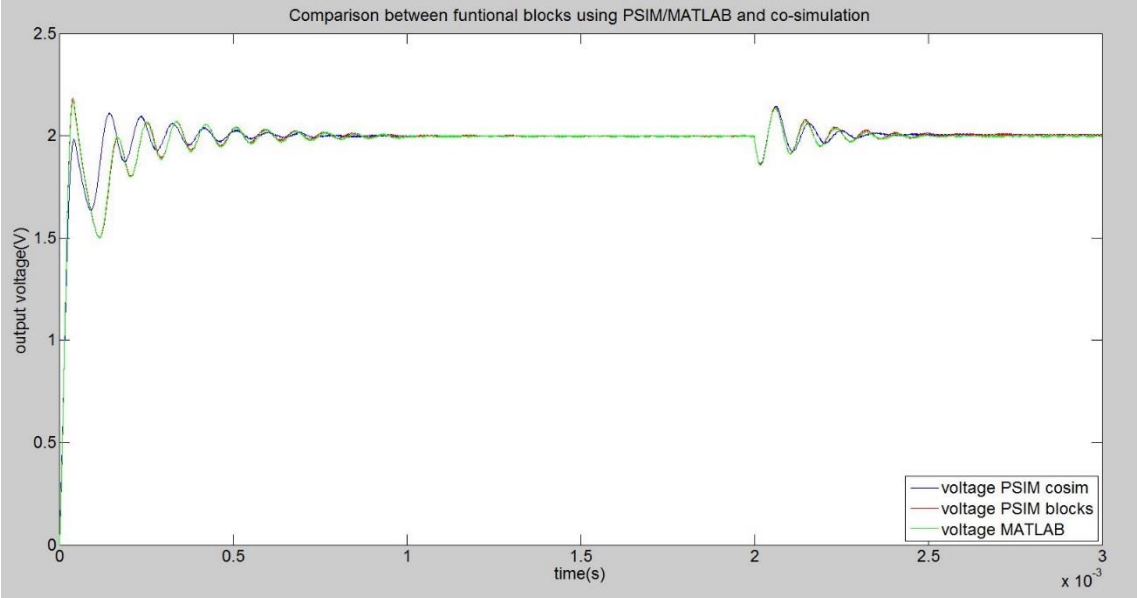


FIGURA 51. RESULTADOS SIMULACIÓN VS CO-SIMULACIÓN ANTE PERTURBACIÓN

Para ver en detalle qué ocurre tras la perturbación se muestra la figura 52.

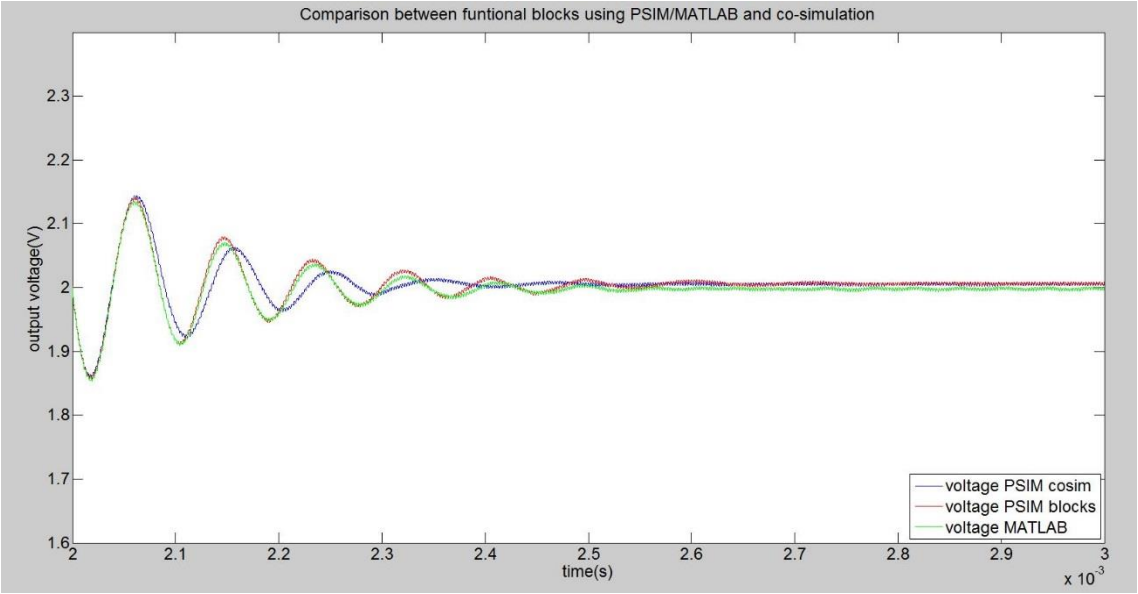


FIGURA 52. RESPUESTA TRAS VARIACIÓN EN LA CARGA 0.4Ω A 0.3Ω

Vemos como al comienzo de la perturbación los tres sistemas son similares. Sin embargo, tras la primera oscilación la respuesta de la co-simulación se desfasa respecto de los modelos. Tal y como parecía, el sistema hardware presenta una respuesta más rápida de lo esperado, lo que implica que se han modificado las características dinámicas del sistema.

5.3.2. Comentarios acerca de los resultados

Tras analizar los resultados obtenidos en el apartado 5.1, se puede considerar equivalente el modelo realizado en MATLAB respecto del modelo de PSIM. Esto demuestra que el diseño en bloques funcionales es independiente de la herramienta que se utilice y por tanto, que el diseño en bloques funcionales es adaptable a otras herramientas.

Por otra parte, los resultados comparativos entre los modelos y la co-simulación muestran una variación en las prestaciones dinámicas. Esto provoca que los tiempos de estabilización y las sobreoscilaciones sean distintas. Sin embargo, las variaciones no son lo suficientemente significativas como para considerar un problema en la implementación.

Además, las características en régimen permanente son similares. Se observa que los retrasos debidos a la implementación digital están correctamente definidos, ya que no existe ningún desfase entre las señales. El valor final está dentro de los márgenes válidos.

5.4. Validación experimental del diseño implementado en la FPGA

5.4.1. Validaciones unitarias de los bloques del sistema en lazo abierto

Con el fin de evitar posibles fallos se proponen una serie de validaciones previas de cada uno de los bloques por separado. Este método de pruebas unitarias permite validar cada bloque de forma independiente y facilita las labores de depuración.

Para este sistema se van a llevar a cabo pruebas sobre el convertidor, el bloque pre-amplificador y el conversor analógico-digital.

5.4.1.1. Convertidor reductor

El convertidor reductor requiere de una señal PWM de disparo para los mosfet y de una tensión de entrada. Además es necesario conectar una resistencia de pull-up a VCC en la señal INH para deshabilitarla.

En la figura 53 puede verse el montaje del convertidor en lazo abierto utilizando un generador de tensión para la entrada y la deshabilitación de la señal INH y un generador de funciones para introducir el PWM.

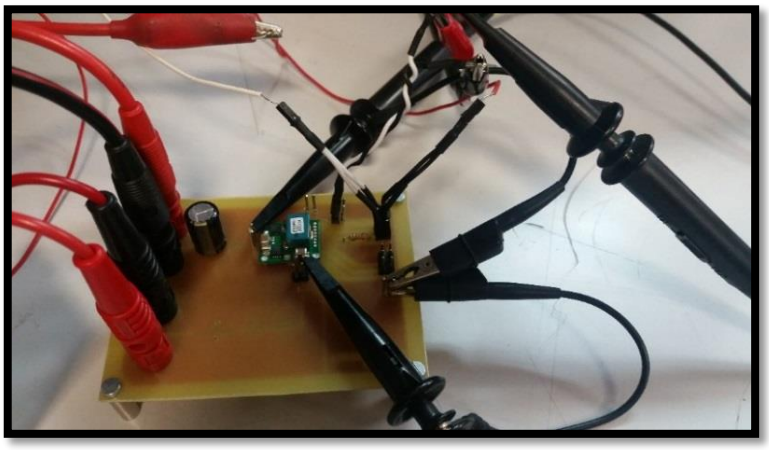


FIGURA 53. MONTAJE CONVERTIDOR LAZO ABIERTO

Los resultados obtenidos pueden verse en la figura 54. La tensión de entrada se muestra en el CH1, donde se aprecian 6V. El CH2 muestra la tensión de salida, que indica 2V. Y el CH3 muestra la señal PWM de disparo con un ciclo de trabajo del 33%.

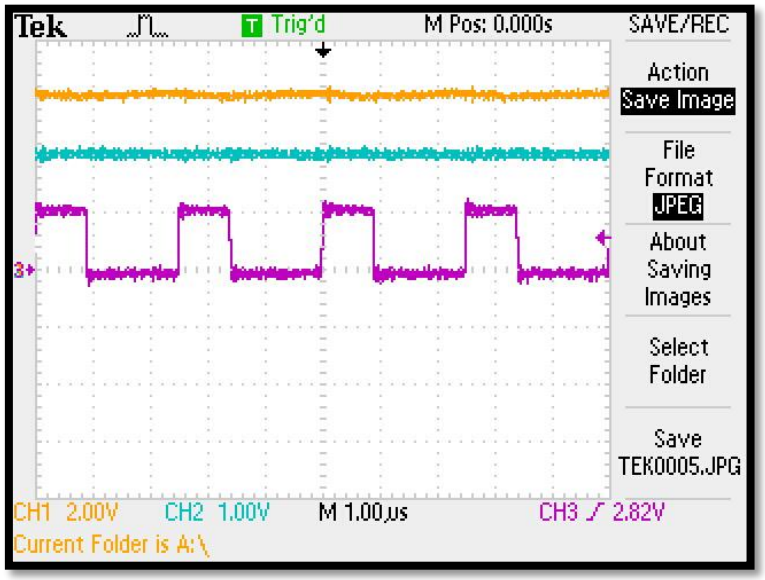


FIGURA 54. SEÑALES OSCILOSCOPIO CONVERTIDOR EN LAZO ABIERTO

Por lo tanto, a la vista de los resultados, concluimos que el convertidor funciona de forma correcta.

5.4.1.2. Bloque pre-amplificador

Para la validación del bloque pre-amplificador se va a hacer uso de los LED e interruptores que hay disponibles en la placa de la FPGA. Haciendo unos pequeños reajustes en el VHDL se configuran los puertos de tal forma que seamos capaces de ver que ganancia carga el amplificador en cada instante.

Como ya indicamos en el capítulo 4, la ganancia de nuestro diseño debe ser -1; que corresponde con el vector "00010001". Por lo que los LED's que deberían iluminarse se muestran en la tabla:

LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
0	0	0	1	0	0	0	1

TABLA 11. VALOR DE LA GANANCIA DEL PRE-AMPLIFICADOR

A la vista de la figura 55, tal y como se esperaba, los LED's que se han encendido son los correctos.

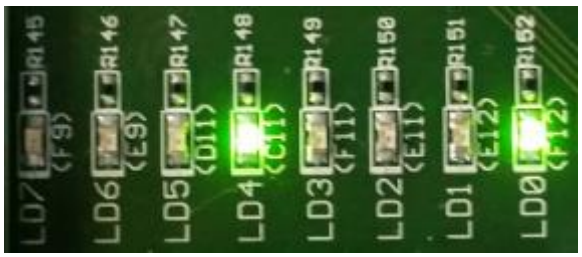


FIGURA 55. VALOR EXPERIMENTAL DE LA GANANCIA DEL PRE-AMPLIFICADOR

5.4.1.3. Adquisición de datos a través del conversor analógico/digital

Al igual que se hizo para el bloque pre-amplificador, se van a utilizar los interruptores y los LED's de la placa para comprobar el funcionamiento del ADC.

La prueba que se va a realizar consiste en introducir por VINA una tensión analógica constante y leer el valor digital obtenido por el canal 0 de la FPGA. Para poder visualizar correctamente qué LED's se encienden y cuáles no, se debe añadir un botón que permita controlar cuando se debe realizar la lectura y actualizar el dato. Debido a que el ADC tiene 14 bits y solo disponemos de 8 LED's se ha decidido configurar los interruptores de tal forma que seamos capaces de leer la parte alta del vector con una configuración, y la parte baja con otra. Los LED's 1 y 0 no se usan cuando estamos viendo la parte baja del vector.

Ya que el generador de tensión tiene un rizado considerable se han tomado 5 muestras para cada valor de entrada. De esta forma es posible ver como los bits más significativos no cambian, pero los menos significativos sí.

- En la primera lectura la tensión de entrada es de 2V tal y como muestra la figura 56.

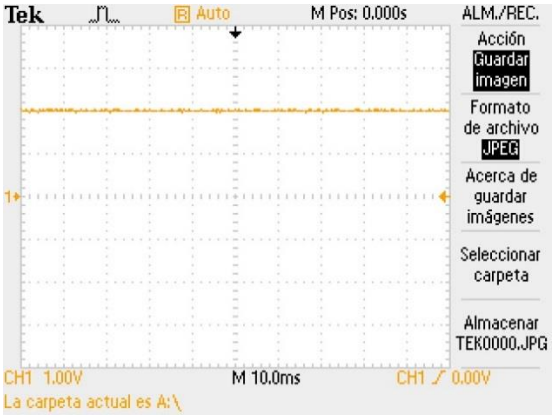


FIGURA 56. TENSIÓN DE 2 VOLTIOS ENTRADA ADC

Para el valor de la figura anterior, los valores registrados por el ADC son (tabla 12):

SW0 = '1' SW1 = '0'								SW0 = '1' SW1 = '1'					
LD7	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0**	**LD7**	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**
Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	1	0	1	1	1	1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1	0	0	0	1	0	0
1	1	0	1	1	1	1	1	0	0	0	0	1	1
1	1	0	1	1	1	1	1	0	0	0	1	1	0
1	1	0	1	1	1	1	1	0	0	1	0	0	1

TABLA 12. VALORES EXPERIMENTALES REGISTRADOS POR EL ADC PARA 2V

En la figura 57 puede verse el resultado de una de las muestras tomada directamente en los LED's de la FPGA, más concretamente la muestra 5, resaltada en amarillo:

FIGURA 57. RESULTADO OBTENIDO POR LOS LED'S PARA TENSIÓN DE ENTRADA 2V

Si despejamos de la ecuación 6 el valor de tensión de entrada podemos comprobar si los resultados son correctos.

$$V_{IN} = -\left(\frac{D \cdot 1{,}25}{8192}\right) - 1{,}65 \qquad (34)$$

Muestra	D	VIN [V]
1	-2113	1,9724
2	-2110	1,9719
3	-2109	1,9718
4	-2106	1,9714
5	-2103	1,9708
Promedio		**1.9718**

TABLA 13. RESULTADO CAPTURA DATO ADC PROMEDIO PARA 2V

El valor promedio obtenido es correcto si tenemos en cuenta que la resolución del ADC, como calculamos en el capítulo 4, tiene un valor de 20mV. Lo que justifica que la desviación entre el valor de entrada y el dato convertido este en torno a esta magnitud. Además, el rizado de tensión de la fuente de pruebas también debe considerarse como fuente de error.

- Para el segundo caso la tensión de entrada es de 1,6V como puede verse en la figura 58. Los resultados obtenidos en la FPGA se muestran en la figura 59:

FIGURA 58. TENSIÓN DE 1,6V ENTRADA ADC

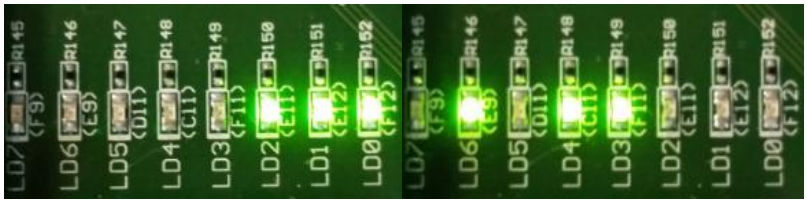


FIGURA 59. RESULTADO OBTENIDO POR LOS LED'S PARA TENSIÓN DE ENTRADA 1.6V

Si usamos de nuevo la tabla anterior obtenemos lo siguiente:

SW0 = '1' SW1 = '0'								SW0 = '1' SW1 = '1'					
LD7	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0**	**LD7**	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**
Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	0	0	1	1	1	0	1	0	1	1	0

TABLA 14. VALORES EXPERIMENTALES REGISTRADOS POR EL ADC PARA 1,6V

Haciendo uso de la ecuación 6 obtenemos el valor de D y podemos calcular el valor de tensión de entrada como muestra la siguiente tabla. De nuevo si consideramos la tensión de resolución del ADC vemos que el resultado concuerda con lo esperado.

Muestra	D	VIN [V]
1	470	**1,5783**

TABLA 15.RESULTADO CAPTURA DATO ADC PARA 1,6V

Con el fin de probar el ADC en todo el rango de medida, además de los ejemplos ya comentados, se van a probar tensiones de entrada de 2,3V, 0,405V y 2,85V.

- Para el caso de 2.3V los resultados obtenidos son:

SW0 = '1' SW1 = '0'								SW0 = '1' SW1 = '1'					
LD7	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0**	**LD7**	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**
Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	1	0	0	0	0	0	0	0	1	1	0	1	1
1	1	0	0	0	0	0	0	1	1	0	0	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1
1	1	0	0	0	0	0	0	1	0	0	0	1	0
1	1	0	0	0	0	0	0	1	1	0	0	0	1

TABLA 16.VALORES EXPERIMENTALES PARA 2,3V

Muestra	D	VIN [V]
1	-4069	2,2708
2	-4044	2,2671
3	-4055	2,2687
4	-4062	2,2698
5	-4047	2,2675
Promedio		**2.2688**

TABLA 17. RESULTADO CAPTURA DATO ADC PARA 2,3V

En los dos siguientes casos, como ya se ha comentado, se va a probar cómo funciona el ADC en regiones próximas a los límites.

- Para 0,405V obtenemos:

SW0 = '1' SW1 = '0'								SW0 = '1' SW1 = '1'					
LD7	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0**	**LD7**	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**
Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1

TABLA 18.VALORES EXPERIMENTALES PARA 0,405V

Muestra	D	VIN [V]
1	8191	0,400
2	8191	0,400
3	8191	0,400
4	8191	0,400
5	8191	0,400
Promedio		**0,400**

TABLA 19. RESULTADO CAPTURA DATO ADC PARA 0,405V

- Y para el último caso, el de 2,85V, se han obtenido los siguientes resultados:

SW0 = '1' SW1 = '0'								SW0 = '1' SW1 = '1'					
LD7	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0**	**LD7**	**LD6**	**LD5**	**LD4**	**LD3**	**LD2**
Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	0	0	0	0	1	1	0	0	0	0	0	1	1
1	0	0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	0	0	1	1	0	0	1	1	0	1	0
1	0	0	0	0	1	1	0	0	1	1	0	1	0
1	0	0	0	0	1	1	0	0	1	0	1	1	0

TABLA 20.VALORES EXPERIMENTALES PARA 2,85V

Muestra	D	VIN [V]
1	-7805	2,841
2	-7782	2,8374
3	-7781	2,8373
4	-7784	2,8377
5	-7792	2,8389
Promedio		**2,8385**

TABLA 21. RESULTADO CAPTURA DATO ADC PARA 2,85V

Por lo tanto, a la vista de los resultados obtenidos en los distintos casos, el bloque de conversión analógico-digital queda completamente validado.

5.4.2. Validación del sistema completo en lazo cerrado

5.4.2.1. Respuesta del sistema en régimen permanente

En este apartado se van a desarrollar los resultados obtenidos tras la implementación del sistema en lazo cerrado. Como ya se comentó en el capítulo 2, la metodología PowerDigit no había sido validada experimentalmente, por lo que uno de los objetivos de este trabajo consiste en cerrar la aplicación de esta metodología en un diseño específico.

Para validar el sistema completo es necesario conectar el convertidor a la placa FPGA a través de las entradas analógicas de las que dispone y la salida DPWM a la entrada de los dispositivos de disparo del convertidor. A continuación se muestra una imagen del sistema completo durante la realización de las pruebas.

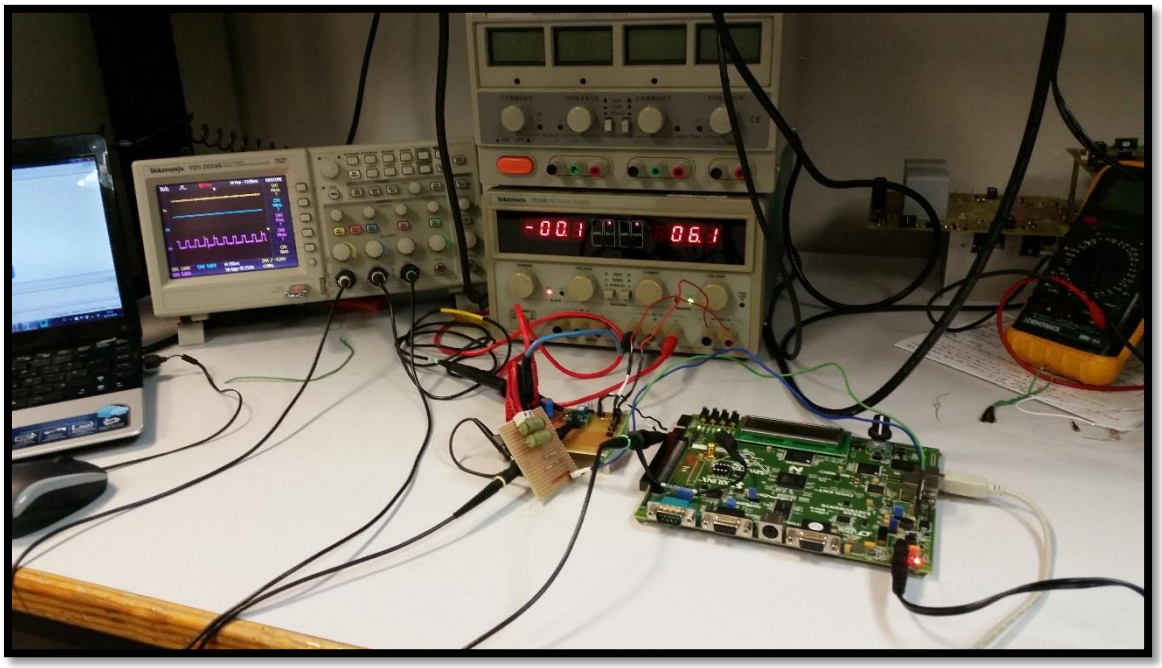


FIGURA 60. SISTEMA DE TEST PARA VALIDACIÓN DEL SISTEMA EN LAZO CERRADO

Los elementos del sistema, así como los aparatos de medida y generación empleados para realizar las pruebas son los siguientes:

- Convertidor Reductor PTD08A10W.
- FPGA Spartan 3E Starter Kit.
- Generador de tensión Tektronix PS280 DC Power Supply.
- Osciloscopio Tektronix TDS 2024B de 4 canales.
- XILINX ISE Design Suite v14.7.

Para llevar a cabo el arranque del convertidor con éxito sin necesidad de implementar un soft-start (arranque suave), ha sido necesario reducir el limitador superior del filtro Anti-Windup a un valor del 50% de ciclo de trabajo. Este valor no afecta a las pruebas realizadas ya que en régimen permanente el ciclo de trabajo es inferior al de este limitador.

En la figura 61 se muestra una captura del osciloscopio en la que pueden verse las tres señales más destacadas del sistema:

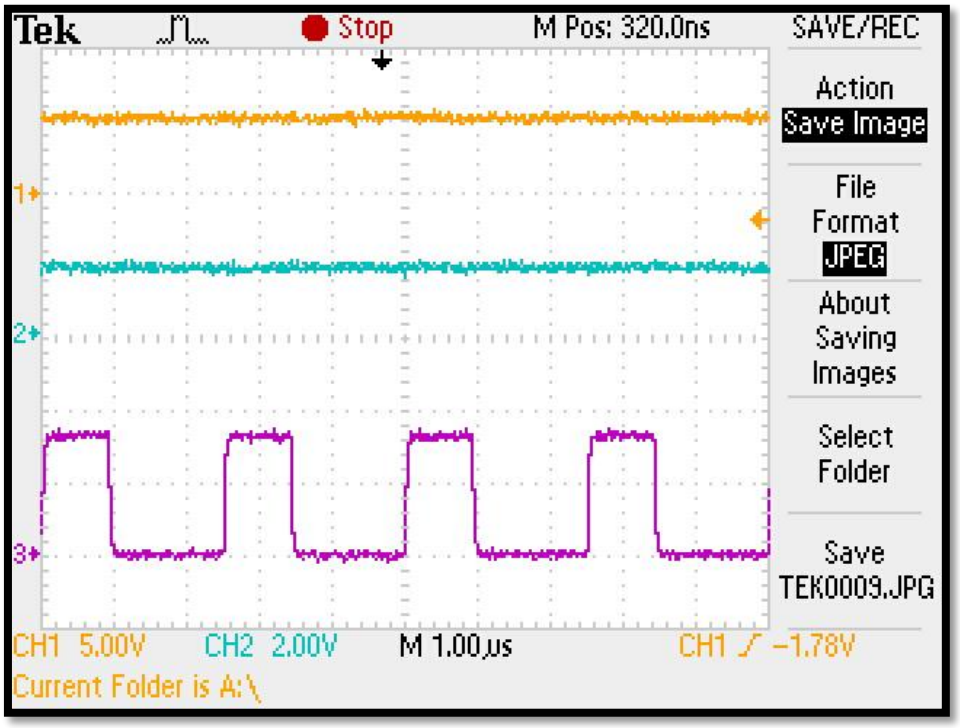


FIGURA 61. RESULTADOS EXPERIMENTALES SISTEMA EN LAZO CERRADO

La señal amarilla, del CH1, representa la tensión de entrada del convertidor que tiene un valor de 6V. La señal morada, CH3, representa la señal DPWM que se obtiene de la FPGA. Y por último, la señal azul, CH2, representa la tensión de salida del convertidor con un valor de 2V.

Si nos fijamos detalladamente en la señal DPWM, vemos como la frecuencia de la señal es de 400kHz, justo la frecuencia de conmutación de las especificaciones mostradas en el capítulo 4, por lo que la señal DPWM se está generando de forma correcta. Para comprobar este resultado se hace uso de la siguiente expresión:

$$f_{DPWM} = \frac{1}{2{,}5div \cdot 1\frac{\mu s}{div}} = \frac{1}{2{,}5\mu s} = 400kHz \qquad (35)$$

Donde *div* representa las divisiones del osciloscopio y f_{DPWM} la frecuencia de la señal DPWM.

Por otra parte, el valor del ciclo de trabajo de la señal morada puede calcularse rápidamente con la ecuación siguiente:

$$d = \frac{t_{on}}{T} = \frac{\frac{4}{5} div \cdot 1 \frac{\mu s}{div}}{2{,}5\mu s} = 0{,}32 \quad (36)$$

Siendo T=2,5μs el período de la señal DPWM y t_{on} el tiempo durante el cual la señal permanece en nivel alto.

Este resultado coincide con los resultados de simulación, por lo que el sistema experimental en lazo cerrado funciona en régimen permanente de forma correcta.

Además, para ver el rizado de la tensión de salida se ha realizado la siguiente captura del osciloscopio (figura 62) donde las señales están de igual forma distribuidas que en la figura 61.

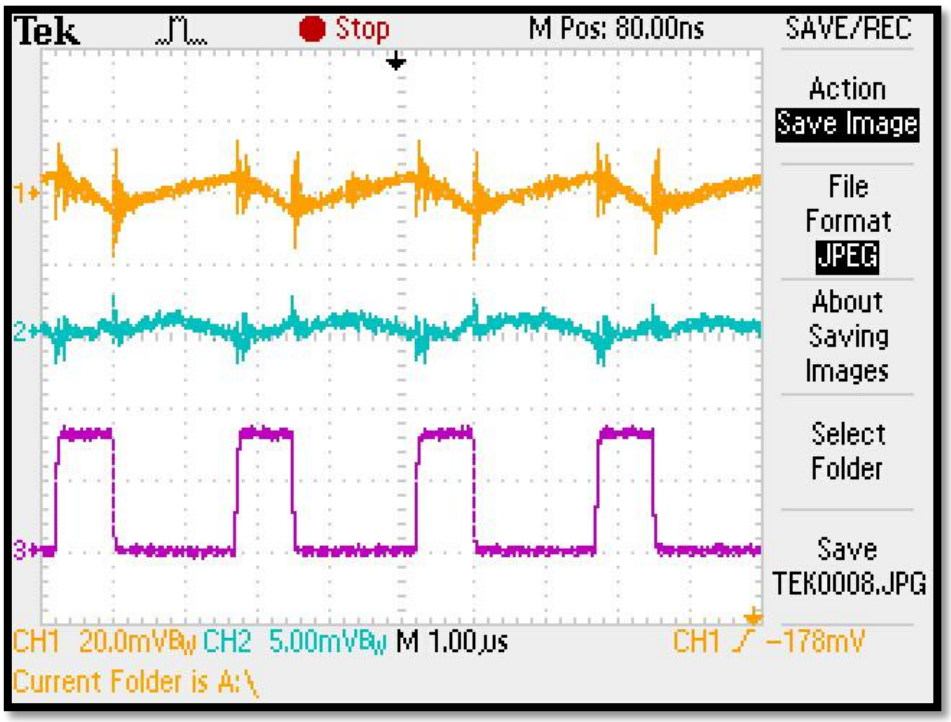


FIGURA 62. VISTA DETALLADA DEL RIZADO DE LA TENSIÓN DE SALIDA

Para calcular el valor del rizado de la tensión de salida se hace uso de la ecuación siguiente:

$$\Delta V_o = \frac{4}{5} div \cdot 5 \frac{mV}{div} = 4mV \quad (37)$$

El valor obtenido concuerda con lo esperado, puesto que en los resultados de co-simulación el rizado de la tensión de salida era de 4mV aproximadamente. Además, la resolución del ADC, 19,6mV es mucho mayor que el resultado obtenido.

Otro detalle que merece la pena destacar son las sobretensiones que sufre la tensión de salida como consecuencia de la conmutación de los MOSFET, que vemos que coincide con los instantes en los cuales la señal DPWM tiene una transición.

En conclusión, los resultados experimentales obtenidos cumplen con las especificaciones iniciales. Tras esta validación se cierra el ciclo de diseño del lazo de control digital para el convertidor reductor habiendo aplicado la metodología PowerDigit.

6. CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado se detallan las conclusiones derivadas de la realización del presente trabajo. Además se incluye una breve descripción sobre los trabajos futuros que permitirían mejorar la metodología PowerDigit.

Para realizar este apartado se tienen en cuenta los objetivos iniciales definidos en el apartado 1.2, y los resultados obtenidos a lo largo de todo el capítulo 5.

El objetivo principal del trabajo consistía en extender y aplicar la metodología PowerDigit en el diseño de un lazo de control digital para un convertidor CC-CC y realizar las validaciones necesarias con el fin de eliminar las limitaciones impuestas inicialmente por la metodología. Tras la realización del presente trabajo se han logrado los siguientes hitos:

- Diseño y validación de los bloques funcionales en la herramienta de simulación MATLAB-Simulink.
- Desarrollo de un nuevo modelo funcional de ADC de ventana.
- Desarrollo de un programa C que configura automáticamente el código VHDL a través de plantillas.
- Incorporación de nuevos bloques VHDL al diseño general.
- Validación en co-simulación del nuevo diseño con otro tipo de ADC.
- Validación experimental en lazo cerrado de todo el sistema.

Los resultados que se han obtenido a lo largo del trabajo muestran las siguientes conclusiones:

- El diseño de bloques funcionales es independiente de la plataforma de desarrollo. Lo cual permite utilizar la metodología PowerDigit en otras herramientas de simulación.
- El nuevo modelo de ADC amplía el abanico de ADCs disponibles, permitiendo que los bloques funcionales puedan funcionar con datos digitales en complemento a dos.
- El programa C permite configurar el diseño de forma automática conectando los ficheros de configuración de PSIM con el fichero top del diseño VHDL.
- El usuario de la metodología que quiera diseñar un lazo de control digital no requiere de conocimientos específicos de lenguaje de descripción hardware. Esto es debido a que todo el diseño VHDL es configurable desde PSIM ejecutando el programa desarrollado.
- El sistema de co-simulación presenta pequeñas diferencias en su respuesta dinámica respecto al diseño en bloques funcionales. Sin embargo, para las pruebas realizadas, esta

variación no afecta en la implementación final del diseño. Por otro lado, las características estáticas del modelo funcional y el modelo de co-simulación son similares.

- Por último, los resultados experimentales obtenidos confirman que es posible diseñar un lazo de control digital aplicando la metodología PowerDigit.

Una vez finalizado el proyecto, surgen líneas futuras de trabajo como son:

- Generación automática de código VHDL empleando MATLAB.

- Comparativa de la eficiencia entre el código generado a partir de los bloques funcionales PSIM, y el generado a través de MATLAB.

- Obtención de la respuesta en frecuencia del sistema para comprender con más profundidad el comportamiento del mismo y poder corregir cualquier diferencia respecto al modelo de simulación.

7. PRESUPUESTO Y PLANIFICACIÓN

7.1. Presupuesto

En este apartado se muestra información detallada del coste total derivado de realizar el proyecto, desglosado en gasto de materiales, software y personal.

PRESUPUESTO GENERAL

SUBAPARTADO MATERIALES

Concepto	Unidades	Precio (€/ud)	Precio total (€)
Convertidor PTD08A010W	1	27,00	27,00
FPGA Spartan-3E Starter Kit	1	150,00	150,00
Resistencia 500kΩ ¼ W	1	0,02	0,02
Resistencia 1kΩ ¼ W	1	0,02	0,02
Resistencia 1Ω 25 W	2	1,93	3,86
Protoboard con soldadura	1	3,19	3,19
		TOTAL SUBAPARTADO MATERIALES	**184,09**

SUBAPARTADO SOFTWARE

Concepto	Licencias (ud)	Precio (€/licencia)	Precio total (€)
PSIM v9.3.4	1	2.000,00	2.000,00
SmartCtrl v1.0	1	500,00	500,00
ModelSim v10.0b	1	100,00	100,00
MATLAB R2013a	1	6.000,00	6.000,00
		TOTAL SUBAPARTADO SOFTWARE	**9.500,00**

SUBAPARTADO RECURSOS HUMANOS

Concepto	Número de horas	Precio (€/hora)	Precio total (€)
Diseño	50	25,00	1.250,00
Validación	150	25,00	3.750,00
Documentación	100	15,00	1.500,00
		TOTAL SUBAPARTADO RR.HH	**6.500,00**
		TOTAL PRESUPUESTO	**16.184,09**
		I.V.A (21%)	**3.398,66**
		TOTAL	**19.582,75**

7.2. Planificación

Una planificación temporal derivada de realizar el proyecto puede verse en la siguiente tabla. Los meses se dividen en quincenas para concretar las fechas en la mayor medida posible. El tiempo medio semanal dedicado a realizar tareas del trabajo ha sido de 12 horas. Además de información detallada de los tiempos de realización de cada tarea se muestra información relativa a los materiales y programas software empleados.

Nº	ACTIVIDAD	RECURSOS		ABRIL		MAYO		JUNIO		JULIO		AGOSTO		SEPTIEMBRE	
		Material	Sofware	1-15	15-30	1-15	15-31	1-15	15-30	1-15	15-31	1-15	15-31	1-15	15-30
1	**ESTUDIO PREVIO**														
1.1	Estudio de la metodología PowerDigit	*PC*	-												
2	**DISEÑO**														
2.1	Modificaciones del lazo de control	*PC*	*ISE Design Suite*												
2.3	Desarrollo programa generación de código automático	*PC*	*DevC++*												
2.2	Diseño del modelo MATLAB	*PC*	*MATLAB*												
3	**VALIDACIÓN**														
3.1	Validación sistema MATLAB	*PC*	*MATLAB, PSIM*												
3.2	Validación co-simulación vs. modelos	*PC*	*MATLAB, PSIM, ModelSim*												
3.3	Implementación física en la FPGA	*FPGA, PC*	*ISE Design Suite*												
3.4	Validación experimental unitaria	*Convertidor, FPGA, Osciloscopio, Generador tensión, Generador funciones*	*ISE Design Suite*												
3.5	Validación sistema en lazo cerrado	*Convertidor, FPGA, Osciloscopio, Generador tensión*	-												
4	**MEMORIA**	*PC*	*Microsoft Office*												

8. BIBLIOGRAFÍA

[1] Granados Gil A. *"Aportaciones a la metodología de diseño de un lazo de control digital para un convertidor de potencia",* Tesis fin de Máster, Universidad Carlos III de Madrid, 2014.

[2] Bibian S., Jin H., *"A Simple Prediction Technique for the Compensation of Digital Control Time Delay in DC Switchmode Power Supplies",* Applied Power Electronics Conference and Exposition, Marzo 1999, vol.2.

[3] Patella B. J., Prodic A., Zirger A., Maksimovic D., *"High-frequency digital controller IC for DC/DC converters"*, Applied Power Electronics Conference and Exposition (APEC), Marzo 2002, vol. 1.

[14] Xiao J., Peterchev A. V., Sanders S. R., *"Architecture and IC implementation of a digital VRM controller"*, Power Electronics Specialists Conference (PESC), Junio 2001, vol. 1.

[5] Boudreaux R. R., Nelms R. M., Hung J. Y., *"Simulation and modeling of a DC-DC converter controlled by an 8-bit microcontroller",* Applied Power Electronics Conference and Exposition (APEC), Febrero 1997, vol.2.

[6] Guo L., Hung J. Y., Nelms R. M. W., *"PID controller modifications to improve steady-state performance of digital controllers for buck and boost converters",* Applied Power Electronics Conference and Exposition (APEC), Marzo 2000, vol. 1.

[7] Peterchev A. V. and Sanders S. R., *"Quantization resolution and limit cycling in digitally controlled PWM converters",* IEEE Trans. Power Electron., vol. 18, no. 1, Enero 2003.

[8] Entrena L., Portela M. *"Diseño de Circuitos Integrados"*, Grado en Ingeniería Electrónica Industrial y Automática. Universidad Carlos III de Madrid, Leganés, 2008. http://www3.uc3m.es/reina/Fichas/Idioma_1/223.14043.html. Última consulta realizada: 20-Sept-2015

[9] XILINX®. *"Digital Clock Manager (DCM) Module"*, Product Specification, Abril 2009.

[10] XILINX®. *"Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs"*, Application Note, Enero 2006.

[11] Granados Gil A. *"Diseño e implementación en FPGA del lazo de control de un convertidor de potencia"*, Trabajo fin de Grado, Universidad Carlos III de Madrid, Septiembre 2012.

[12] XILINX®. *"Spartan-3E FPGA Starter Kit Board User Guide"*, Datasheet, Enero 2011.

[13] Texas Instruments®. *"PTD08A010W"*, Datasheet, Febrero 2010.

[14] Barrado Bautista A. *"Sistemas Electrónicos de Potencia"*, Grado en Ingeniería Electrónica Industrial y Automática. Universidad Carlos III de Madrid, Leganés, 2015. http://gsep.uc3m.es/html/index.html. Última consulta realizada: 17-Ago-2015

[15] Barrado Bautista A., Lázaro Blanco A. *"Problemas de electrónica de Potencia"*, Pearson Prentice Hall, 2007. ISBN: 9788420546520.

[16] PowerSim Inc. *"ModCoupler-VHDL User's Guide"*, Version 1.0, Mayo 2011.

ANEXOS

A.1. Anexo I: Manual de utilización de la herramienta ModCoupler

ModCoupler es un módulo disponible en PSIM que permite comunicarse con ModelSim. Esto significa que es posible realizar una co-simulación de la parte de potencia, que está en PSIM, con la parte del algoritmo de control digital, que está en ModelSim. La figura 63 muestra la arquitectura del sistema [16].

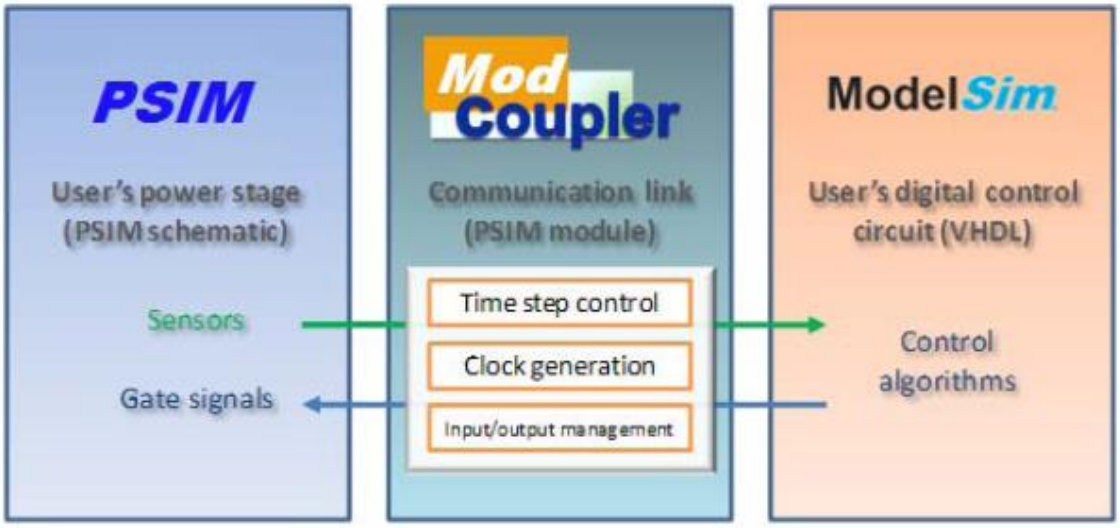


FIGURA 63. ARQUITECTURA DE CO-SIMULACIÓN [7]

A.1.1. Configuración de la estructura de directorios

Para poder utilizar el bloque ModCoupler es necesario llevar a cabo una serie de pasos previos. Ya que no existe literatura que presente detalladamente estos pasos, y con el fin de facilitar la labor de los diseñadores se ha decidido desarrollar este manual.

En primer lugar, cuando vayamos a utilizar un diseño que contenga el bloque ModCoupler es recomendable establecer una estructura de ficheros determinada. En este manual se presenta una posible solución, aunque no es la única opción posible. A continuación, se listan los pasos a realizar para mantener la estructura de ficheros aquí propuesta.

1. Crear una carpeta de proyecto que contenga el esquemático de PSIM, *".psimsch"*.
2. Crear una subcarpeta que contenga los subcircuitos que utiliza el esquemático principal y los ficheros *".txt"* con los parámetros genéricos que utiliza el diseño.
3. Crear una subcarpeta con el nombre *"vhdl"* que contenga todos los ficheros *".vhd"* que utiliza el diseño; y el fichero de compilación *"compile.bat"*.

4. Por último, si se desea almacenar información de alguna simulación. Crear una subcarpeta que contenga los ficheros *".smv"* y/ o *".txt"* que se generan al guardar una simulación.

La carpeta que contiene los diseños VHDL deberá tener un fichero *".vhd"* que será la entidad de más alto nivel en el diseño. Un ejemplo válido sería el siguiente:

```
entity modcoupler is
    port ( clk   : in  std_logic;
           reset : in  std_logic;
           vo    : in real;
           dpwm  : out  std_logic);
end modcoupler;
```

FIGURA 64. EJEMPLO ENTIDAD MODCOUPLER

Para la entidad top existen algunas limitaciones en su diseño:

- El reloj de entrada deberá llamarse obligatoriamente *"CLK"*. Cualquier otro nombre para el reloj no será detectado por el bloque ModCoupler de PSIM.
- Los tipos de datos que admite la entidad son: bit, std_logic, real, integer, bit_vector y std_logic_vector.

Estas limitaciones son las únicas que deberán tenerse en cuenta, el resto del diseño puede abordarse con total normalidad.

A.1.2. Configuración del fichero de compilación

El fichero *"compile.bat"* es editable y contiene un script en lenguaje TCL que sirve para compilar todos los bloques que se utilizarán en la co-simulación. Un ejemplo de este fichero totalmente configurado se muestra en la figura 65:

```
rem vdel -all
vlib work

@echo off
set VHDLDIR=.
vcom %VHDLDIR%\control_loop.vhd
vcom %VHDLDIR%\control_adc.vhd
vcom %VHDLDIR%\adc.vhd
vcom %VHDLDIR%\temp.vhd
vcom %VHDLDIR%\Modulador.vhd
vcom %VHDLDIR%\Regulador.vhd
vcom %VHDLDIR%\adcPSIM.vhd
vcom %VHDLDIR%\ModCoupler.vhd
vcom %VHDLDIR%\control_compensator.vhd
rd /S /Q "../work"
move /Y work ..

pause
```

FIGURA 65. EJEMPLO CONFIGURACIÓN FICHERO COMPILE.BAT

El diseñador deberá modificar el fichero incorporando los nombres de los bloques que vaya a utilizar. Una vez el fichero esté configurado, se deberá ejecutar. Tras la ejecución se creará automáticamente una subcarpeta con el nombre *"work"* dentro de la carpeta de proyecto creada en el punto 1.

La figura 66 muestra la pantalla que debería verse tras la compilación de los ficheros. En caso de existir un fallo en alguno de los ficheros, el compilador nos dará información de qué bloque ha fallado y en qué línea se encuentra el fallo.

Un error muy habitual consiste en localizar el fichero *"compile.bat"* en un directorio distinto a los *".vhd"* que queremos compilar, es por ello que se hace hincapié en mantener una estructura ordenada.

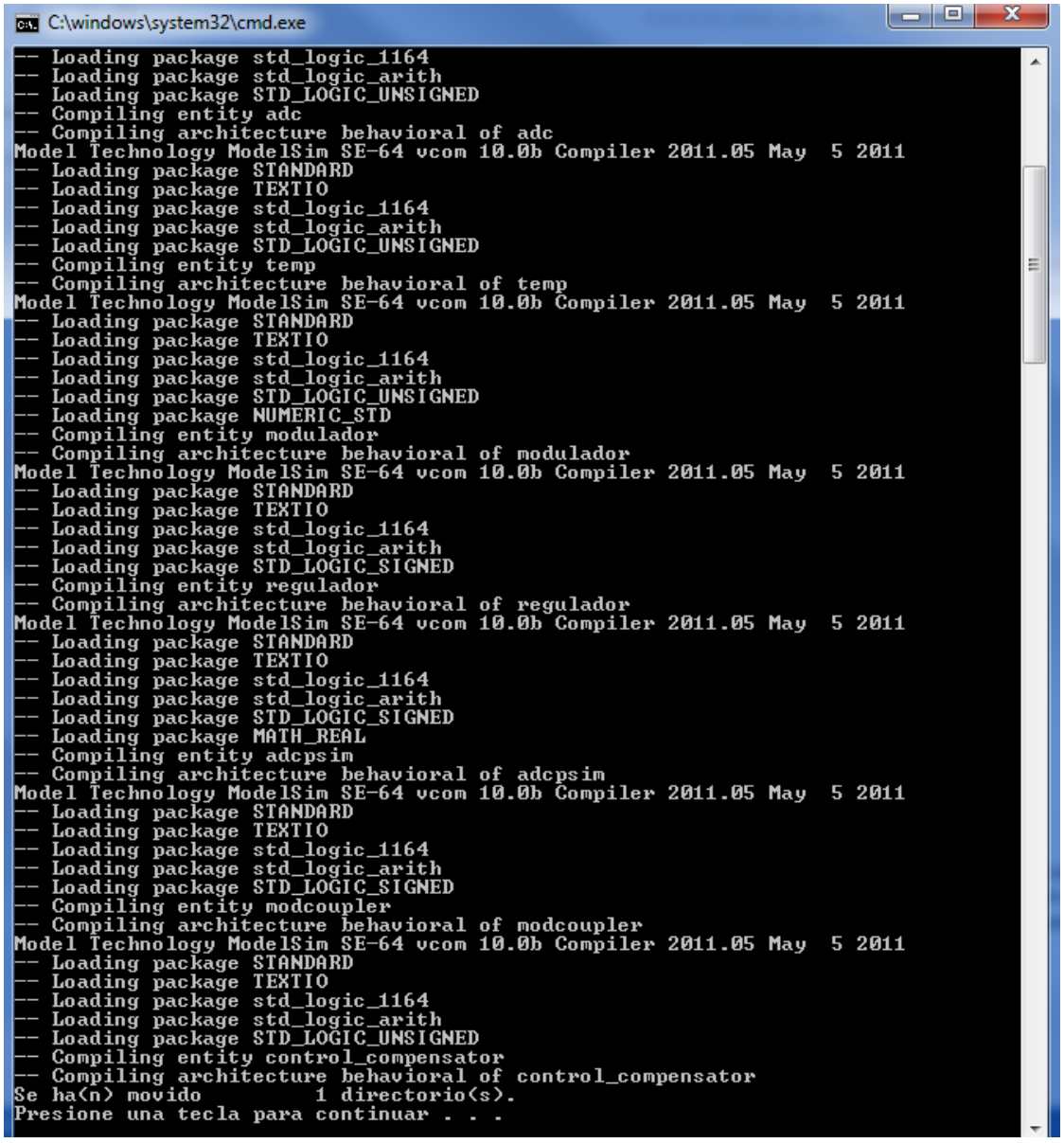


FIGURA 66. PANTALLA DE COMPILACIÓN

A.1.3. Configuración parámetros del bloque ModCoupler

Finalizado el proceso de compilación, sólo quedaría configurar el bloque ModCoupler de PSIM introduciendo los parámetros del bloque que hay en la figura 67.

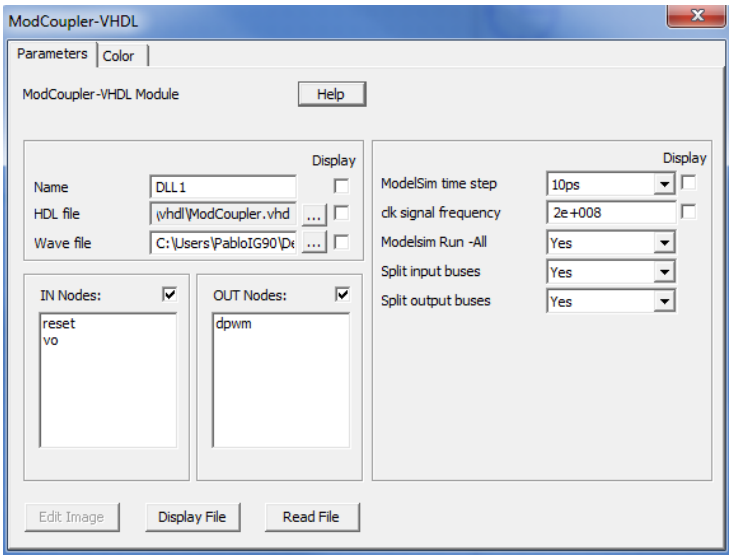


FIGURA 67. PARÁMETROS DE CONFIGURACIÓN DEL BLOQUE MODCOUPLER

- **HDL file**. Este parámetro requiere la ruta hasta el fichero top del diseño. Como ya se vio, será siempre el *"ModCoupler.vhd"*. Es importante seguir bien los pasos anteriores y definir adecuadamente este fichero ya que es el que servirá como interfaz directa entre el simulador PSIM y el resto del código.
- **Wave file**. Es un fichero de extensión *".do"* que contiene información de las señales que desea representar el diseñador. No es necesario configurar este parámetro ni tampoco crear el fichero de formas de onda, pero se recomienda su uso para tener una batería de señales a representar de forma automática sin tener que escogerlas cada vez que se realice una nueva co-simulación. Incluso, se recomiendo crear varios *".do"* distintos que permitan realizar comprobaciones por bloques, teniendo un test mucho más organizado.
- **IN/OUT nodes**. Una vez se introduzca la ruta del fichero ModCoupler y se pulse el botón *ReadFile* se rellenarán las dos listas con los puertos definidos en la entidad *"ModCoupler.vhd"*, a excepción del CLK.
- **ModelSim time step**. Este parámetro representa el paso de simulación de ModelSim. Es necesario combinar los time step de PSIM y ModelSim con el fin de que no existan conflictos entre los simuladores. La única condición que debe reunir el time step de ModelSim es que debe ser menor que el time step de PSIM.
- **CLK signal frecuency**. Se introducirá la frecuencia de reloj de entrada al ModCoupler.
- **ModelSim Run –All**. Siempre y cuando tengamos un fichero *".do"* asignado en el Wave file, seleccionar *"Yes"* cargará automáticamente la lista de señales y sólo será necesario actualizar la lista para comenzar a ver los resultados. Y seleccionar *"No"* nos obliga a pulsar manualmente la opción Run –All dentro de ModelSim para comenzar la simulación.

- **Split input/output buses**. Sirve para dividir el bloque de señales de entrada/salida en elementos individuales, con el fin de tratar cada puerto entrada/salida individualmente. Por defecto esta opción aparece como *"Yes"*.

Los botones de *Display File* y *Read File* sirven para mostrar el fichero *".vhd"* top y recargar la lista de entradas/salidas respectivamente. En la figura 68 puede verse un ejemplo de montaje de un sistema con ModCoupler:

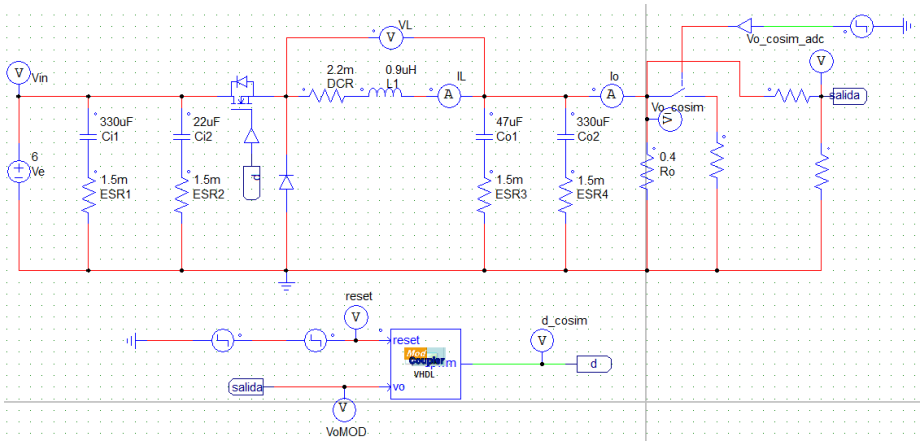


FIGURA 68. EJEMPLO DE PLANTA CONECTADA AL BLOQUE MODCOUPLER

A.1.3.1. Simulación del diseño

Una vez configurado el bloque, lo siguiente será simular el diseño. Tras pulsar el botón *Run Simulation* de PSIM, transcurren entre 15-40 segundos hasta que se abre de forma automática la pantalla de ModelSim. Una vez abierta, en caso de tener habilitada la opción *ModelSim Run –All* se cargará el fichero con las formas de onda. De lo contrario se deberá pulsar manualmente el botón de simulación. Finalmente, durante la simulación podremos ver las señales internas que deseemos. Un ejemplo puede verse en la figura 69.

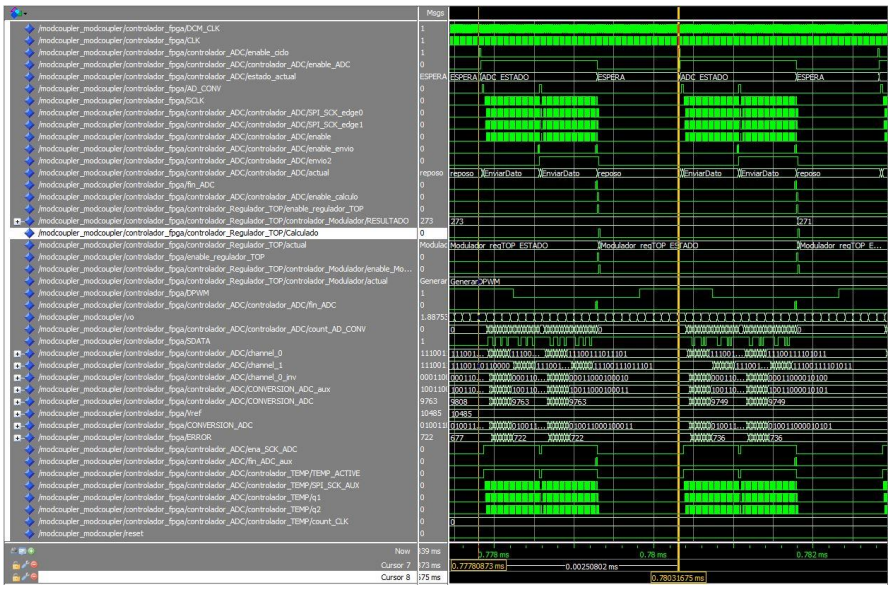


FIGURA 69. VISUALIZACIÓN DE SEÑALES EN MODELSIM

A.1.4. Problemas derivados del uso de ModCoupler

Pese a ser una potente y eficaz herramienta de co-simulación, existen algunos problemas que podemos encontrarnos al utilizar ModCoupler. Algunos de estos problemas que se han detectado pueden retrasar los plazos del diseñador ya que no hay literatura disponible para solucionarlos. Por ello, en este apartado se comentarán las situaciones más habituales de fallo.

- Una de las cuestiones más habituales que puede causar mayor problemática es elegir qué versiones de PSIM y ModelSim se deben usar. En principio, no debería existir ningún problema en usar cualquier versión de PSIM que disponga de ModCoupler y cualquier versión de ModelSim. Sin embargo, tras varias pruebas se ha llegado a la conclusión de que la mejor combinación posible es PSIM 9.3.4 / ModelSim 10.0b o posteriores. Otro factor a tener en cuenta es que ambas versiones sean de 32 o 64 bits; alternar entre una y otra da lugar a problemas.
- Cuando realizamos una co-simulación y la pantalla de ModelSim nunca llega a generarse saltará un fallo en PSIM de que no ha sido posible conectar las herramientas. Para solucionar este problema se deben cerrar los procesos de PSIM abiertos con el administrador de tareas y volver a ejecutarlos de nuevo.
- Siempre que se realizan modificaciones en el diseño es necesario recompilar el fichero *"compile.bat"*. De no ser así no es posible actualizar los cambios. Se recomienda pulsar el botón *Read File* tras cada nueva compilación, pues así nos aseguramos de que capture bien el fichero *ModCoupler*.
- Cuando sea necesario salir de la co-simulación antes de llegar al final, es obligatorio detenerla pulsando *STOP* antes de cerrar la pantalla de ModelSim. De lo contrario, en la siguiente co-simulación no conectará las herramientas debido a que se quedan procesos internos colgados. Esto obligará a reiniciar los programas como ya se comentó en el segundo punto.

Estos fallos son los más habituales y los que han sido detectados tras realizar diversas pruebas. Sin embargo, pueden existir otros problemas derivados de ModCoupler que no estén descritos en este manual.